



Essential Studio 2013 Volume 4 - v.11.4.0.26

Essential HTMLUI for Windows Forms



Contents

1	Overview	9
1.1	Introduction To Essential HTMLUI	9
1.2	Prerequisites and Compatibility	10
1.3	Documentation	11
2	Installation and Deployment	13
2.1	Installation.....	13
2.2	Sample and Location.....	13
2.3	Deployment Requirements	16
2.3.1	Toolbox Entries.....	16
2.3.2	DLLs	16
3	Getting Started	17
3.1	Lesson 1: Creating an HTML Display Application	18
3.1.1	Displaying HTML By Using the HTMLUI Control	18
3.2	Lesson 2: Creating an HTML Layout.....	21
3.2.1	Creating the User Interface	21
3.2.2	Creating And Displaying Custom Controls	25
4	Concepts And Features	30
4.1	Loading HTML	31
4.1.1	Loading As a Startup Document	31
4.1.1.1	Startup File Sample	33
4.1.2	Loading At Run Time.....	34
4.1.2.1	Loading the File From Disk	36
4.1.2.1.1	Load File From Disk Sample	37
4.1.2.2	As Links From One HTML Document To Another	37
4.1.2.2.1	File Links Sample	39
4.1.2.3	Loading the File From URI	40
4.1.2.4	Loading HTML Which Is In the Form Of Text.....	41
4.1.2.5	Load the File from Resource	41
4.1.2.5.1	Load Resource File Sample	45
4.2	MHT Formats.....	45

4.3	Control Events	47
4.3.1	HTMLUI Control Events Sample	52
4.4	Element Events	53
4.4.1	HTMLUI Bubbling Events Sample	56
4.4.2	HTMLUI Element Events Sample.....	57
4.5	HTML Elements	58
4.5.1	Element Types.....	59
4.5.2	A - Anchor Element	59
4.5.3	B - Bold Element.....	60
4.5.4	BODY Element	60
4.5.5	BR - Break Element.....	61
4.5.6	CODE Element	61
4.5.7	CUSTOM Element.....	61
4.5.8	DIV - Division Element.....	62
4.5.9	EM - Emphasize element	62
4.5.10	FONT Element.....	62
4.5.11	FORM Element.....	62
4.5.12	H1 - H6 Header Elements	62
4.5.13	HEAD Element	63
4.5.14	HTML Element.....	63
4.5.15	HR - Horizontal Rule Element	63
4.5.16	I - Italics Element	63
4.5.17	IMG - Image Element	63
4.5.18	INPUT Element.....	64
4.5.19	LI - List Element.....	65
4.5.20	LINK Element	65
4.5.21	OL - Ordered List Element	66
4.5.22	P - Paragraph Element.....	66
4.5.23	PRE - Preformatted Element.....	66
4.5.24	SCRIPT Element	66
4.5.25	SELECT Element	67
4.5.26	SPAN Element.....	68
4.5.27	STRONG Element.....	68
4.5.28	STYLE Element.....	68
4.5.29	TABLE Element.....	69

4.5.30	TD - Table cell Element.....	69
4.5.31	TEXTAREA Element	70
4.5.32	TH - Table Head Element.....	70
4.5.33	TR - Table Row Element	70
4.5.34	UL - Unordered List Element.....	71
4.5.35	U - Underline Element	71
4.6	HTML Tags.....	71
4.6.1	A - Anchor Tag	72
4.6.2	ABBR - Abbreviation Tag	72
4.6.3	Acronym Tag	73
4.6.4	Comment Tag.....	73
4.6.5	Font Style Tags	74
4.6.6	BODY - Body Tag.....	75
4.6.7	BR - Break Tag.....	75
4.6.8	DIV - Division Tag.....	76
4.6.9	FORM - Form Tag	76
4.6.10	HEAD - Head Tag.....	77
4.6.11	H1 - H6 Header Tag	78
4.6.12	HR - Horizontal Rule Tag	78
4.6.13	HTML - HTML Tag.....	79
4.6.14	IMG - Image Tag	80
4.6.15	INPUT - Input Tag	80
4.6.16	LI - List Item Tag.....	82
4.6.17	LINK - Link Tag.....	82
4.6.18	OL - Ordered List Tag.....	83
4.6.19	OPTION - Option Tag.....	84
4.6.20	P - Paragraph Tag	84
4.6.21	PRE - Preformatted Tag.....	85
4.6.22	SCRIPT - Script Tag.....	86
4.6.23	SELECT - Select Tag	87
4.6.24	SPAN - Span Tag	88
4.6.25	STYLE - Style Tag.....	88
4.6.26	SUB - Subscript Tag.....	90
4.6.27	SUP - Superscript Tag.....	90
4.6.28	TABLE - Table Tag.....	91

4.6.29	TD - Table Cell Tag	92
4.6.30	TEXTAREA - Text Area Tag	93
4.6.31	TH - Table Header Tag.....	94
4.6.32	TITLE - Title Tag.....	95
4.6.33	TR - Table Row Tag	96
4.6.34	UL - UnOrdered List Tag	96
4.6.35	HTML Tags Sample.....	97
4.7	Custom Controls.....	98
4.7.1	Custom Controls Sample.....	100
4.8	HTML Forms.....	101
4.8.1	HTMLUIForms Sample.....	102
4.9	HTML Bookmarks.....	103
4.9.1	HTMLUI Bookmarks Sample	104
4.10	HTML Tables	105
4.10.1	HTMLUI Tables Sample	106
4.11	HTML Layout	107
4.11.1	HTMLUI Chat Sample	108
4.12	HTML Renderer.....	108
4.12.1	HTMLUI Browser Sample.....	110
4.12.2	HTMLUI Editor Sample.....	111
4.13	Style Sheets CSS	112
4.13.1	Cascading Style Sheets	113
4.13.1.1	Inline StyleSheet.....	114
4.13.1.2	Internal StyleSheet	114
4.13.1.3	External StyleSheet.....	115
4.13.1.3.1	Design Time	116
4.13.1.3.2	Run Time.....	116
4.13.1.4	Class Selectors.....	118
4.13.1.4.1	Name Class Selectors.....	118
4.13.1.4.2	ID Class Selectors.....	118
4.13.1.5	CSS Comments.....	119
4.13.2	Supported CSS Attributes	120
4.13.2.1	Background - CSS.....	120
4.13.2.1.1	Background Image	120
4.13.2.1.2	Background Color.....	120

4.13.2.1.3	Background Repeat.....	121
4.13.2.2	Text – CSS	121
4.13.2.2.1	Text Color	121
4.13.2.2.2	Text Align	122
4.13.2.2.3	Text Decoration	122
4.13.2.3	Font – CSS	122
4.13.2.3.1	Font Family.....	123
4.13.2.3.2	Font Size	123
4.13.2.3.3	Font Style	123
4.13.2.3.4	Font Weight	124
4.13.2.4	Border – CSS	124
4.13.2.4.1	Border Color	124
4.13.2.4.2	Border Width	125
4.13.2.5	Padding – CSS	125
4.13.2.5.1	Padding	126
4.13.2.5.2	Padding Right.....	126
4.13.2.5.3	Padding Left	126
4.13.2.5.4	Padding Top	126
4.13.2.5.5	Padding Bottom	126
4.13.2.6	Dimension - CSS.....	126
4.13.2.7	Classification - CSS.....	127
4.13.2.8	Positioning - CSS	127
4.13.2.9	Pseudo - Classes	128
4.13.3	HTMLUIUseCSS Sample	128
4.13.3.1	HTMLUIElementsCSS Sample	129
4.14	HTMLUI Appearance.....	130
4.14.1	HTMLUIAppearance Sample	131
4.15	HTML Format	132
4.15.1	HTMLFormat Sample	133
4.16	Element Format	134
4.16.1	ElementFormat Sample.....	136
4.17	Exporting.....	137
4.17.1	HTMLUIExporting Sample.....	139
4.18	Keyboard	139
4.18.1	HTMLUIKeyboard Sample	140

4.19	Printing.....	140
4.19.1	HTMLUIPrinting Sample.....	143
4.20	Searching.....	143
4.20.1	HTMLUISearching Sample.....	144
4.21	Scrolling.....	145
4.21.1	HTMLUIAutoScroll Sample	146
4.22	Scripting.....	146
4.22.1	HTMLUIScripting Sample.....	147
4.23	Text Selection.....	148
4.23.1	HTMLUITextSelection Sample	149
5	Frequently Asked Questions	151
5.1	How To Access All the Child Elements Of an HTML Element In the HTMLUI Control?	151
5.2	How To Access the HTML Elements Loaded Into the Control?	153
5.3	How To Access the Inner HTML Text Of the Current HTML Element In the HTMLUI Control?.....	154
5.4	How To Access the Name Of an HTML Element At Run-time?.....	155
5.5	How To Add an Attribute To an HTML Element And Change Its Value At Run-time?	156
5.6	How To Change a Characteristic Of an HTML Element Before It Is Displayed?	157
5.7	How To Change the Default Font Used For Rendering the HTML Document In the HTMLUI Control?.....	160
5.8	How To Delete an HTML Element From a Document Loaded Into the HTMLUI Control At Run-time?	161
5.9	How To Enable the HTMLUI Control To Load HTML Documents That Have Been Dragged Onto the Control?	162
5.10	How To Enable User Interaction With the HTML Elements	164
5.11	How To Get an Object For the Control Present In an HTML Element In the HTMLUI Control?.....	167
5.12	How To Load Custom Controls As Part Of the HTML Layout?	168
5.13	How To Load HTML Into the HTMLUI Control?	171
5.13.1	Loading As Startup Document	172
5.13.2	Loading At Run Time.....	173
5.14	How To Print the Contents Of the HTMLUI Control?	175
5.15	How To Specify a CSS File For HTML Content?	175
5.16	How To Toggle the Visibility Of an HTML Element In the HTMLUI control At Run-time?	176

1 Overview

This section covers information on the HTMLUI control, its key features, prerequisites to use the control, its compatibility with various OS and browsers and finally the documentation details complimentary with the product. It comprises the following sub sections:

1.1 Introduction To Essential HTMLUI

Essential HTMLUI is a .NET Windows Forms control that supports Hyper Text Markup Language (HTML) rendering. HTML elements are exposed as programmatic elements that support standard events. These elements can be custom-painted. HTMLUI can be used as a HTML viewer or to easily lay out and customize rich application interfaces.

This control will benefit those users who want to build flexible applications. Some good examples are applications that need to be localized, themed; interfaces that need to be updated through the network without distributing the whole application, and so on.

HTMLUI is helpful in creating cool UI applications like Chat windows. It can be used to create a multiple dialog application and it can be used as an UI Editor.

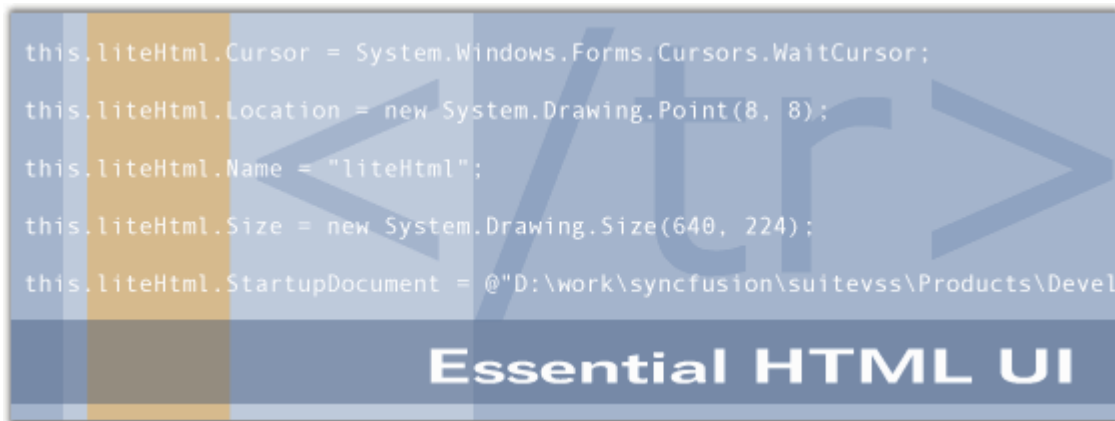


Figure 1: Essential HTML UI

Key Features

Some of the key features of Essential HTMLUI are listed below:

- Essential HTMLUI uses the unique Syncfusion **Flex-Layout** mechanism to lay out any kind of user interface in a flexible manner. Defining the HTML for the layout renders the display with the rules of HTML layout applied automatically. This flexible approach to layout brings several advantages to applications that need their interfaces to be flexible.




- HTMLUI is primarily used as an HTML renderer / viewer. The completely managed browser interface provided by HTMLUI is a natural choice for any HTML rendering interfaces such as custom web browsers, HTML help viewers, email readers, and so on.
- MHTML enables you to send and receive Web pages and other HTML documents using e-mail programs. MHTML enables you to embed images directly into the body of your e-mail messages rather than attaching them to the message. MHTML uses **MIME (Multipurpose Internet Mail Extensions)** which provides facilities to allow multiple objects in a single Internet e-mail message; represents formatted multifont text messages; represents non-textual material such as images, and so on.

The product comes with numerous samples as well as an extensive documentation to guide you. This User Guide provides detailed information on the features and functionalities of the HTMLUI control. It is organized into the following sections:

- **Overview**-This section gives a brief introduction to our product and its key features.
- **Installation and Deployment**-This section elaborates on the install location of the samples, license etc.
- **Getting Started**-This section guides you on getting started with Windows application, controls etc.
- **Concepts and Features**-The features of Essential HTMLUI are illustrated with use case scenarios, code examples and screen shots under this section.
- **Frequently Asked Questions**-This section discusses various frequently asked questions with their answers illustrated with examples and code snippets.

Document Conventions

The conventions listed below will help you to quickly identify the important sections of information, while using the content:

Convention	Icon	Description
Note	 Note:	Represents important information
Example	Example	Represents an example
Tip		Represents useful hints that will help you in using the controls/features
Additional Information		Represents additional information on the topic

1.2 Prerequisites and Compatibility

This section covers the requirements that are mandatory for using Essential HTMLUI. It also lists operating systems and browsers compatible with the product.

Prerequisites

The prerequisites details are listed below:

Development Environments	<ul style="list-style-type: none">• Visual Studio 2013• Visual Studio 2012• Visual Studio 2010 (Ultimate and Express)• Visual Studio 2008 (Team, Professional, Standard and Express)• Visual Studio 2005 (Team, Professional, Standard and Express)• Borland Delphi for .NET• SharpCode
.NET Framework versions	<ul style="list-style-type: none">• .NET 4.5• .NET 4.0• .NET 3.5• .NET 2.0

Compatibility


The compatibility details are listed below:

Operating Systems	<ul style="list-style-type: none">• Windows 8.1• Windows Server 2008 (32 bit and 64 bit)• Windows 7 (32 bit and 64 bit)• Windows Vista (32 bit and 64 bit)• Windows XP• Windows 2003
-------------------	---

1.3 Documentation

Syncfusion provides the following documentation segments to provide all necessary information for using Essential Tools controls in an efficient manner.

Type of documentation	Location
Readme	[drive:]\Program Files\Syncfusion\Essential

	Studio\X.X.X.X\Infrastructure\Data\Release Notes\readme.htm
Release Notes	[drive:]\Program Files\Syncfusion\Essential Studio\X.X.X.X\Infrastructure\Data\Release Notes\Release Notes.htm
User Guide (this document)	<p>Online</p> <p>http://help.syncfusion.com/resources(Navigate to the HTMLUI for Windows Forms User Guide.)</p> <p> Note: Click Download as PDF to access a PDF version.</p> <p>Installed Documentation</p> <p>Dashboard -> Documentation -> Installed Documentation.</p>
Class Reference	<p>Online</p> <p>http://help.syncfusion.com/resources(Navigate to the Windows Forms User Guide. Select <i>HTMLUI</i> in the second text box, and then click the Class Reference link found in the upper right section of the page.)</p> <p>Installed Documentation</p> <p>Dashboard -> Documentation -> Installed Documentation.</p>

2 Installation and Deployment

This section covers information on the install location, samples, licensing, patches update and updation of the recent version of Essential Studio. It comprises the following subsections:

2.1 Installation

For step-by-step installation procedure for the installation of Essential Studio, refer to the **Installation** topic under **Installation and Deployment** in the **Common UG**.

See Also

For licensing, patches and information on adding or removing selective components, refer the following topics in Common UG under **Installation and Deployment**.

- Licensing
- Patches
- Add / Remove Components

2.2 Sample and Location

This section covers the location of the installed samples and describes the procedure to run the samples through the sample browser. It also lists the location of source code.

Sample Installation Location

The Essential HTMLUI Windows Forms samples are installed in the following location.

...\\My Documents\\Syncfusion\\EssentialStudio\\Version Number\\Windows\\HTMLUI.Windows\\Samples\\2.0

Viewing Samples

To view the samples, follow the steps below:

1. Click **Start-->All Programs-->Syncfusion-->Essential Studio <version number> -->Dashboard**. Essential Studio Enterprise Edition window is displayed.

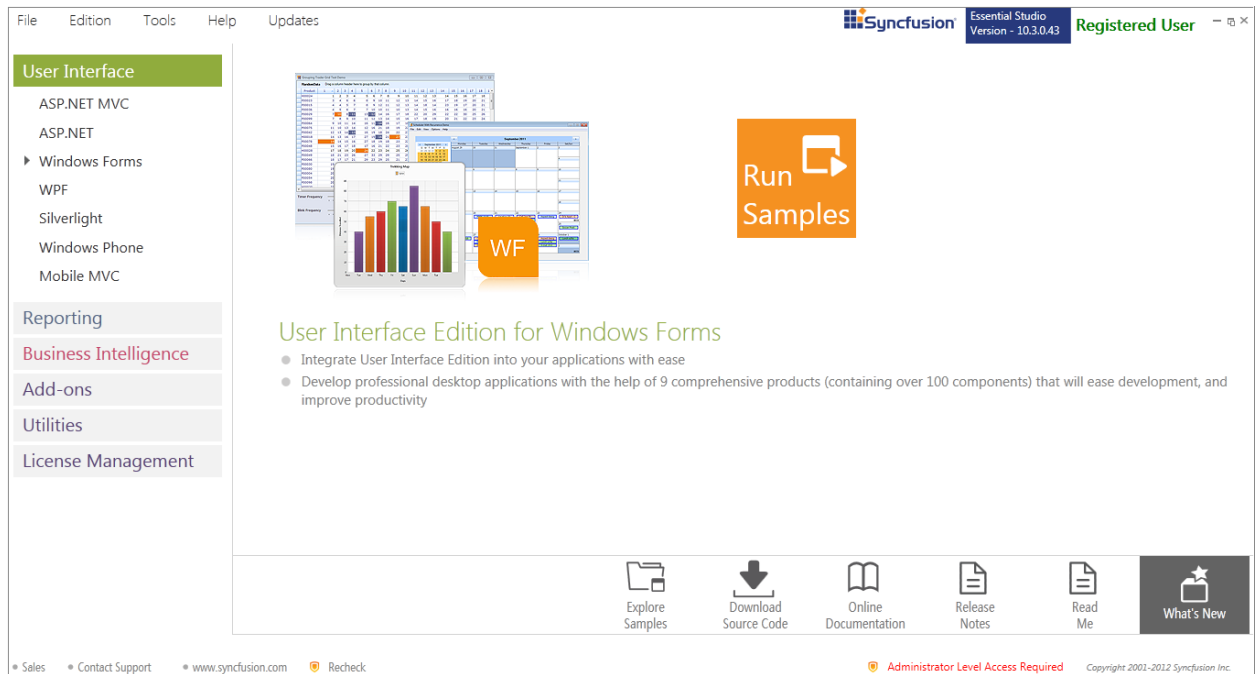


Figure 2: Syncfusion Essential Studio Dashboard

2. In the Dashboard window, click **Run Samples** for Windows Forms under UI Edition. The UI Windows Form Sample Browser window is displayed.



Note: You can view the samples in any of the following three ways:

- **Run Samples**-Click to view the locally installed samples
- **Online Samples**-Click to view online samples
- **Explore Samples**-Explore BI Web samples on disk

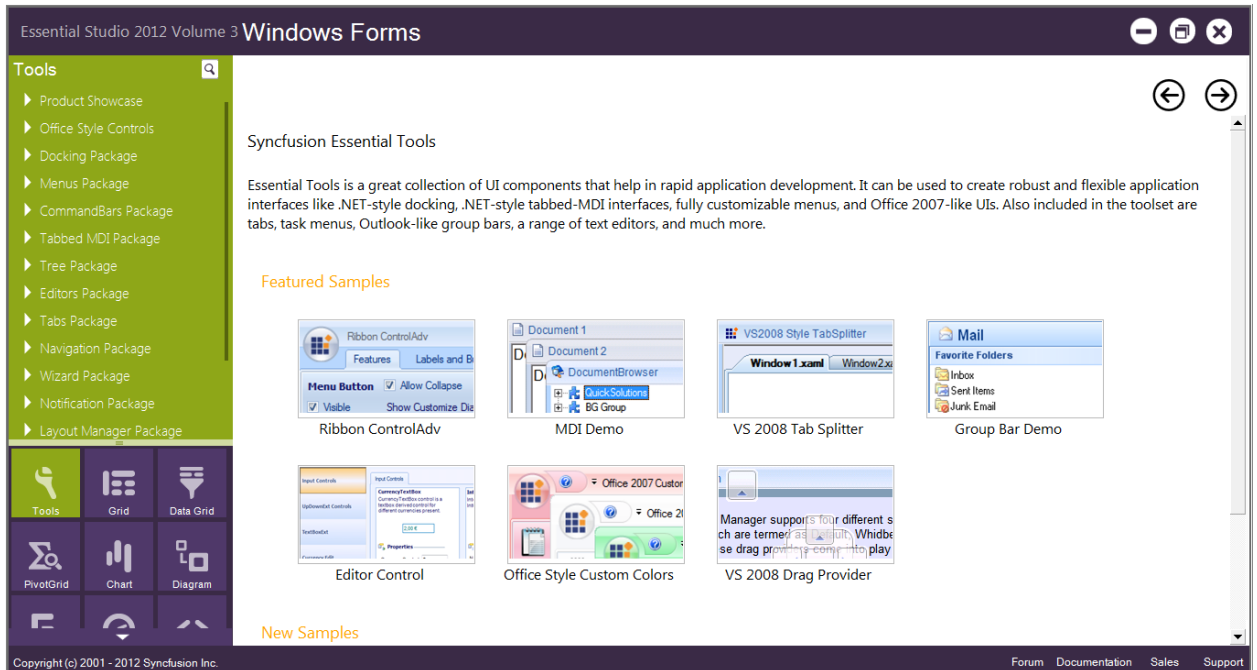


Figure 3: User Interface Edition Windows Forms Sample Browser

3. Click **HTML UI** from the bottom-left pane. HTML UI samples will be displayed.

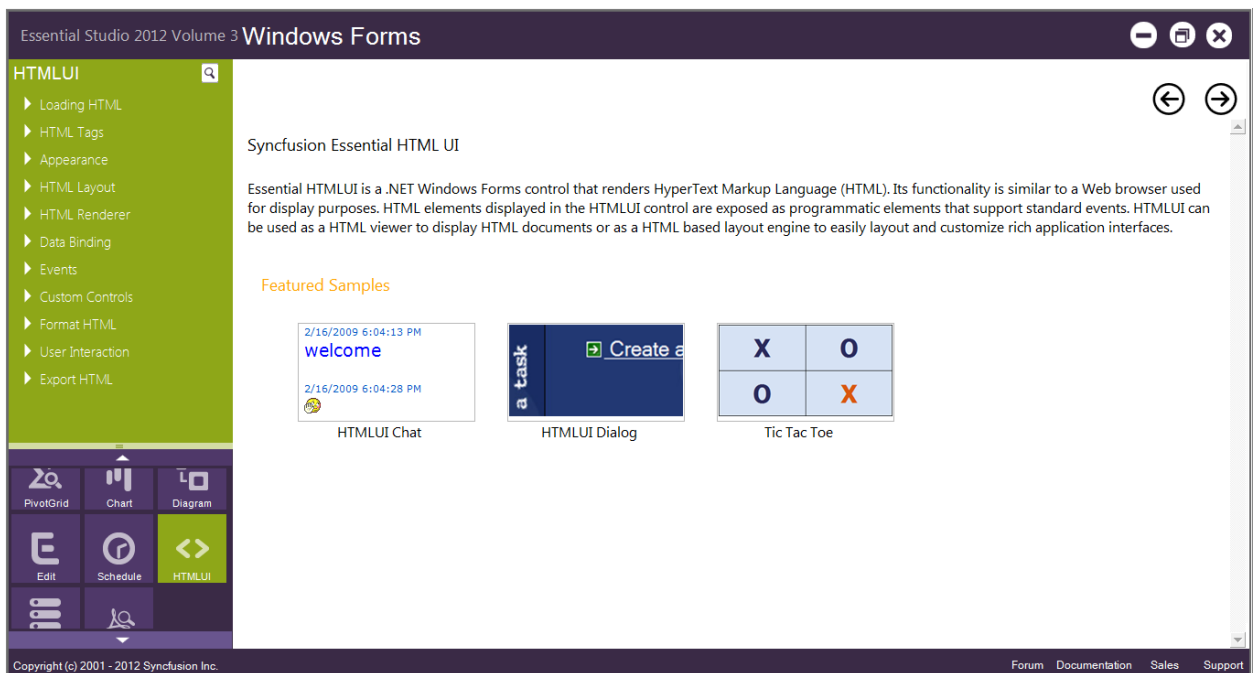


Figure 4: HTMLUI Windows Samples

4. Select any sample and browse through the features.

Source Code Location

The source code for HTMLUI Windows is available at the following default location:

[System Drive]:\Program Files\Syncfusion\Essential Studio\[Version Number]\Windows\HTMLUI.Windows\Src

2.3 Deployment Requirements

This section illustrates the deployment requirements for using Essential HTMLUI in the following topics:

2.3.1 Toolbox Entries

Essential HTMLUI places the following control into your Visual Studio .NET toolbox, from where you can drag each control onto a form and start working with it.


- HTMLUIControl

2.3.2 DLLs

While deploying an application that references a Syncfusion Essential HTMLUI assembly, the following dependencies must be included in the distribution.

Windows Forms – HTMLUI

- Syncfusion.Core.dll
- Syncfusion.Shared.Base.dll
- Syncfusion.Shared.Windows.dll
- Syncfusion.HTMLUI.Base.dll
- Syncfusion.HTMLUI.Windows.dll
- Syncfusion.Scripting.Base.dll
- Syncfusion.MIME.Base.dll

 Syncfusion.HTMLUI.Base.dll depends on Syncfusion.MIME.Base.dll. So this needs to be included in the deployment of any application which uses HTMLUI.

3 Getting Started

What You Will Learn

This tutorial will show you how easy it is to get started using Essential HTMLUI. It will give you a basic introduction to the concepts you need to know before getting started with the product and some tips and ideas on how to implement HTMLUI into your projects to improve customization and increase efficiency. The lessons in this tutorial are meant to introduce you to HTMLUI with simple step-by-step procedures.

Creating an HTML Display Application

[Lesson 1](#) will show you how to load HTML from any source and display it as an HTML display application (like a web browser or an HTML enabled email application).

Creating an HTML Layout

[In Lesson 2](#) you will learn how to lay out your user interfaces using HTMLUI. You will also learn how to let the users interact with the various HTML elements from within your application code.

3.1 Lesson 1: Creating an HTML Display Application

The HTMLUI control can be used for displaying HTML documents with standard HTML / CSS formatting.

In this lesson, you will learn about the following:

3.1.1 Displaying HTML By Using the HTMLUI Control

1. Create a new **Windows Forms** application and open the main form for the application in the designer. Add the Syncfusion controls to your VS.NET toolbox if you haven't done so already. Drag an **HTMLUI control** onto the form.
2. HTML can be loaded into the HTMLUI control from the following sources:
 - From a HTML file
 - From a URI (Uniform resource Identifier)
 - From a Stream
3. Add a **MainMenu** component from the toolbox onto the form. Also add a **OpenFileDialog** component to the form and name it as "openDlg".

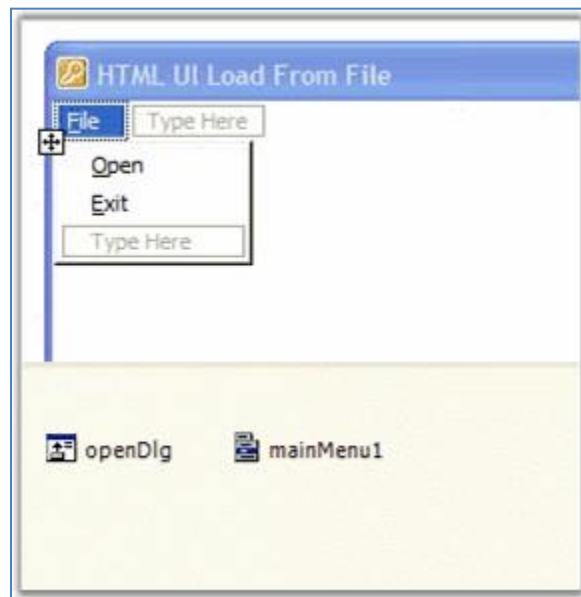


Figure 5: HTMLUI Control and Menu for Loading HTML Files

4. Add a handler for the **Open** menu item by double-clicking on the menu.

[C#]

```
this.menuItem2.Click += new System.EventHandler(this.menuItem2_Click);

private void menuItem2_Click(object sender, System.EventArgs e)
{
    // Gets or Sets the initial directory displayed by file dialog box
    openDlg.InitialDirectory = GetFilesLocation();

    // Gets or Sets the current file name filter string, which determines the
    // choices that appear in the
    // "Save as File Type" or "File of type" box in the dialog box.
    openDlg.Filter = "HTML files (*.htm)|*.htm|HTML Files (*.html)|*.html";
    if( DialogResult.OK == openDlg.ShowDialog() )
    {
        string filePath = openDlg.FileName;
        this.htmluiControl1.LoadHTML(filePath);
    }
}
```

[VB.NET]

```
Me.menuItem2.Click += New System.EventHandler(Me.menuItem2_Click)

Private Sub menuItem2_Click(ByVal sender As Object, ByVal e As
System.EventArgs)

    ' Gets or Sets the initial directory displayed by file dialog box
    openDlg.InitialDirectory = GetFilesLocation()

    ' Gets or Sets the current file name filter string, which determines the
    choices that appear in the
    ' "Save as File Type" or "File of type" box in the dialog box.
    openDlg.Filter = "HTML files (*.htm)|*.htm|HTML Files (*.html)|*.html"
    If DialogResult.OK = openDlg.ShowDialog() Then
        Dim filePath As String = openDlg.FileName
        Me.htmluiControl1.LoadHTML(filePath)
    End If
End Sub
```

5. Now run the sample and try loading a HTML document into the HTMLUI control.



Figure 6: Document Loaded into the HTMLUI Control

Any HTML document can be loaded from a file by using the method shown in this sample.

3.2 Lesson 2: Creating an HTML Layout

The HTMLUI control can be used for displaying sophisticated user interfaces. HTML provides extensive means to layout and customize display elements. The HTMLUI control adds the ability to create user interfaces using HTML from within Windows Forms applications using managed code.

In this lesson, you will learn about about the following:

3.2.1 Creating the User Interface

1. Create a new **Windows Forms** application and open the main form for the application in the designer. Add Syncfusion controls to your **VS .NET** toolbox if you haven't done so already. Drag an **HTMLUI control** onto the form.

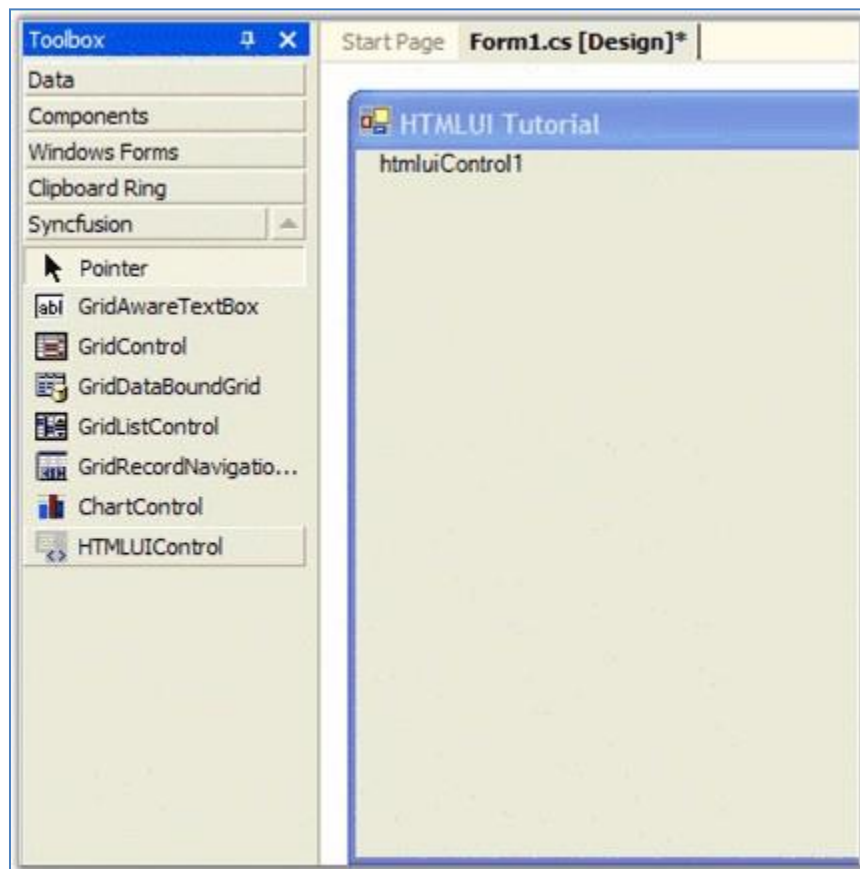


Figure 7: HTMLUI Control

2. Some of the appearance and behavior-related aspects of the HTMLUI control can be controlled by setting the appropriate properties through the **Properties Grid**. The HTMLUI

control displays a title bar at the top of the control. Set the title for this HTMLUI control to be "HTMLUI Tutorial".

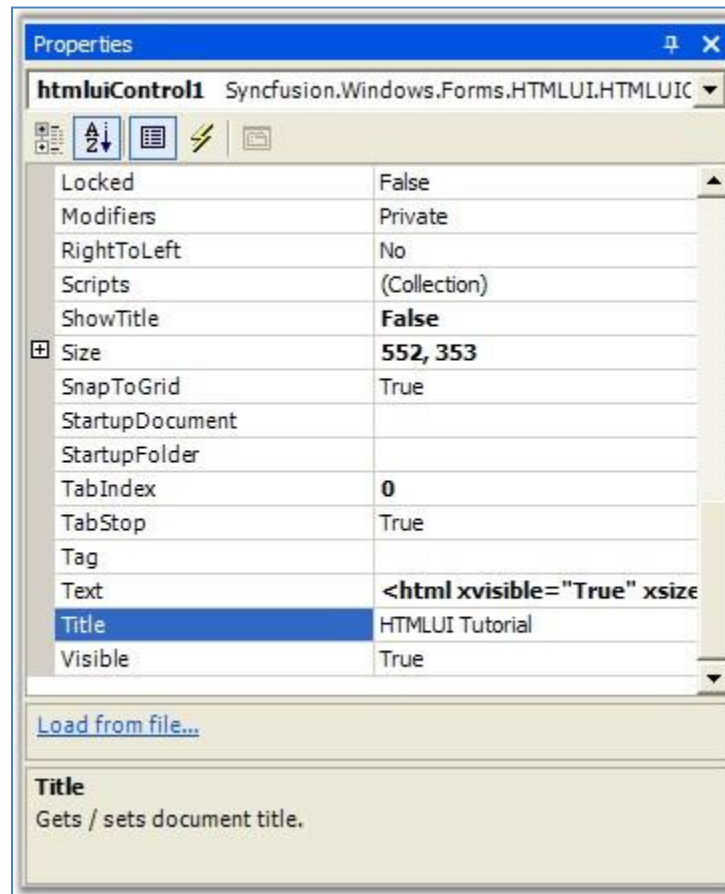


Figure 8: Viewing Properties

3. Add a new HTML document to the Windows Forms project. Edit the HTML document to define the user interface. In this case, you will have an HTML table with 3 rows and 1 column.

[HTML]

```
<HTML>
<HEAD>
<TITLE> Creating User Interface </TITLE>
</HEAD>
<BODY bgcolor="#ffffff">

<TABLE id="Table1" height="360" cellSpacing="1" cellPadding="1" width="392"
border="1">

<TR>
```

```

<TD align="center" height="72" valign="middle">
<INPUT id="txt" type="text" size="40" name="Text1"></INPUT>
</TD>
</TR>

<TR>
<TD align="center" height="209" valign="middle">
<TEXTAREA id="txtArea" name="Textareal" rows="9" cols="35"></TEXTAREA>
</TD>
</TR>

<TR>
<TD align="center" valign="middle">
<INPUT id="btn" type="button" size="" value="Button" name="Button1"></INPUT>
</TD>
</TR>

</TABLE>

</BODY>
</HTML>

```

4. As shown in the HTML document above, a **textbox**, a **textarea** and a **button** control has been added in the HTML document. The objective is to create an user interface by adding a **Click** event to the button element and on clicking the button, the text controls are made to display some text.
5. The **LoadFinished** event is executed when the HTML document is loaded in the HTMLUI control. Add a handler for the LoadFinished event of the HTMLUI control. Access the HTML elements inside the managed code with objects created for each element as defined in the HTMLUI namespace.

```

[C#]

//Objects declaration made global
INPUTElementImpl text;
INPUTElementImpl button;
TEXTAREAELEMENTImpl textArea;

//HTMLUI control LoadFinishedEvent handler declaration
this.htmluiControll1.LoadFinished += new
System.EventHandler(this.htmluiControll1_LoadFinished);

//HTMLUI control LoadFinishedEvent definition
private void htmluiControll1_LoadFinished(object sender, System.EventArgs e)
{
    Hashtable html = this.htmluiControll1.Document.GetElementsByUserIdHash();

    this.text = html["txt"] as INPUTElementImpl;
}

```

```
this.button = html["btn"] as INPUTElementImpl;
this.textArea = html["txtArea"] as TEXTAREAElementImpl;

//Click Event declaration for HTML button element
this.button.Click += new EventHandler(button_Click);
}

private void button_Click(object sender, EventArgs e)
{
    //Click Event definition and Text control UserInterface
    this.text.UserControl.CustomButton.Text = "HTML provides extensive means to
    layout and customize display elements.";
    this.textArea.UserControl.CustomButton.Text = "The HTMLUI control adds the
    ability to create user interfaces using HTML from within Windows Forms
    applications using managed code.";
}
```

6. Now, run the sample which displays the text on clicking the button as shown below.

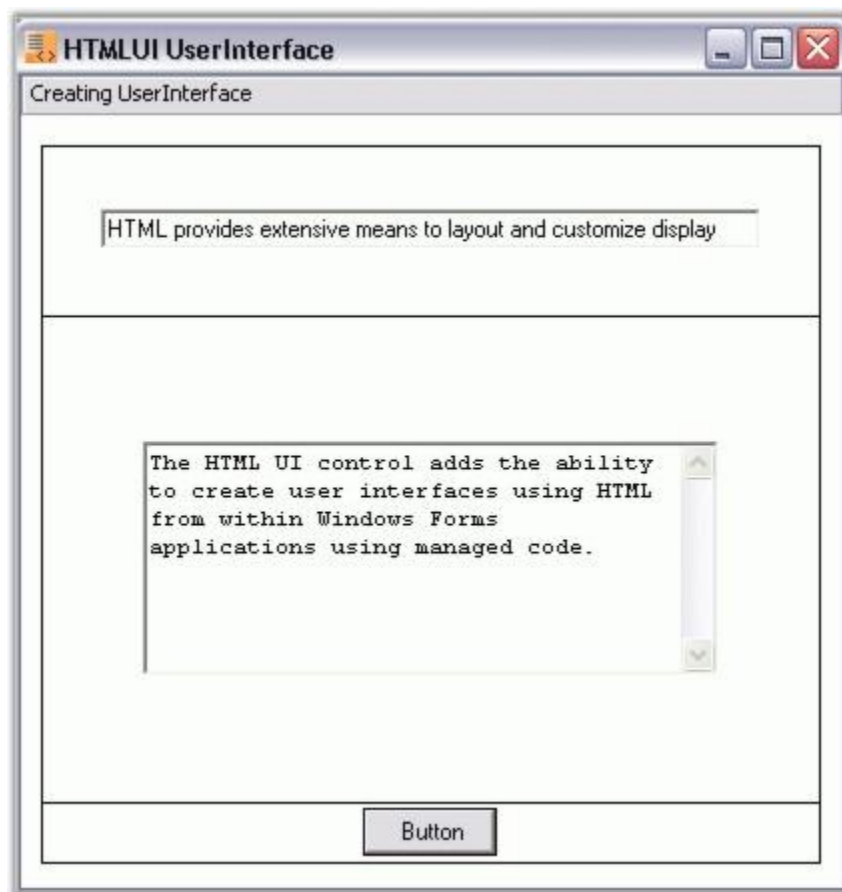


Figure 9: User Interface using HTMLUI

3.2.2 Creating And Displaying Custom Controls

To create and display Custom Controls:

1. Add the **Custom** tags in the HTML document to display custom Windows Forms controls. In this case, a MaskedTextBox, MonthCalendar and DataGrid will be displayed in each of the cells in the HTML Table.

[HTML]

```
<HTML>
<HEAD>
<TITLE>HTMLUI CUSTOM CONTROLS</TITLE>
<style>
.tttDisplay { text-decoration: none; color: #ffffff; font-family: Tahoma; font-
size: 34pt; font-weight: bold; line-height: 30px; padding-left: 2px; }
</style>
</HEAD>
<BODY>

<TABLE id="CustomControls" cellSpacing="0" cellPadding="0" width="100%"
bgColor="silver" border="1" height="100%" align="center">

<TR>
<TD class="tttDisplay" height="33%" width="100%" id="cctd1" vAlign="center">
<maskededittextbox id="maskedEditTextBox1" height="20" width="136">
</maskededittextbox>
</TD>
</TR>

<TR>
<TD class="tttDisplay" height="33%" width="100%" id="cctd2" vAlign="center">
<monthcalendar id="monthCalendar1" width="199" height="155"></monthcalendar>
</TD>
</TR>

<TR>
<TD class="tttDisplay" height="33%" width="100%" id="cctd3" vAlign="center">
<datagrid id="dataGrid1" width="304" height="144"></datagrid>
</TD>
</TR>

</TABLE>

</BODY>
</HTML>
```

Note the custom tags `maskedtextbox`, `monthcalendar`, and `datagrid`. These tags do not have any relation to the name of the control type they represent. Set the desired size of the custom control by setting the **width** and **height** attributes.

In the previous step, three custom controls were defined as part of the HTMLUI interface. The actual Windows Forms controls that represent these definitions have to be added to the form.

2. Drag a **MonthCalendar** control from the toolbox and drop it on the form. The **MonthCalendar** will be named `monthCalendar1` by default. Use this name in the next step to associate it with the appropriate HTML-defined control.
3. Drag a **MaskedTextBox** control and a **DataGrid** control onto the form.

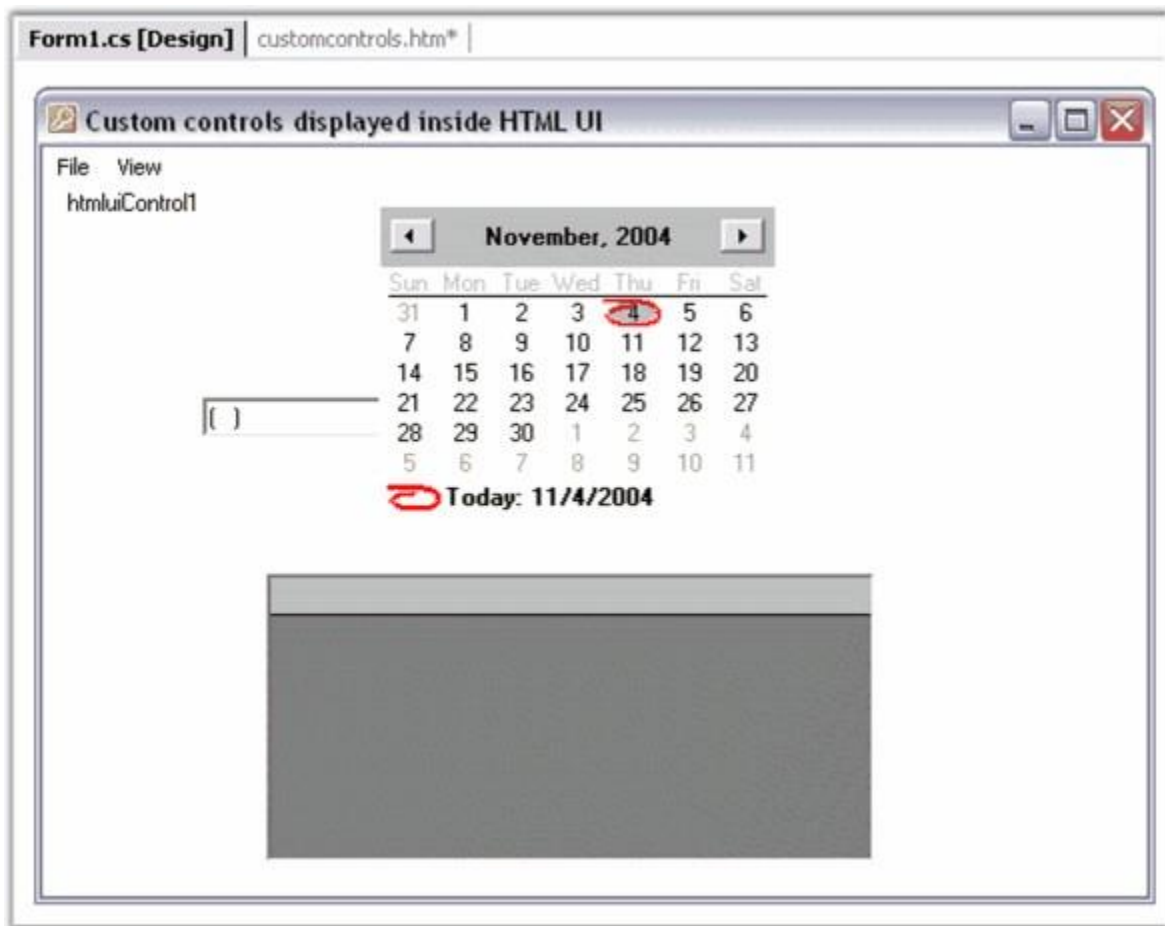


Figure 10: Custom Controls

4. Add a handler for the **PreRenderDocument** event of the **HTMLUI** control.

This step associates the Windows Forms controls on the form with controls defined in the HTML. This is done in the `PreRenderDocument` event handler of the HTMLUI control. This event is raised before the control renders the HTML elements (after it has parsed the HTML document into HTML elements).

[C#]

```
this.htmluiControll1.PreRenderDocument += new
Syncfusion.Windows.Forms.HTMLUI.PreRenderDocumentEventHandler(this.htmluiContro
ll1_PreRenderDocument);

// Event that is to be raised when a tree of element has been created and their
size and location have
// not been calculated yet.
private void htmluiControll1_PreRenderDocument(object sender,
Syncfusion.Windows.Forms.HTMLUI.PreRenderDocumentArgs e)
{
    Hashtable htmelements = new Hashtable();
    htmelements = e.Document.ElementsByUserID;

    BaseElement maskedEditTextBoxElement1 = htmelements["maskedEditTextBox1"] as
    BaseElement;

    // Create a new Wrapper object
    new CustomControlBase( maskedEditTextBoxElement1, this.maskedEditBox1 );

    BaseElement monthCalendarElement1 = htmelements["monthCalendar1"] as
    BaseElement;
    new CustomControlBase( monthCalendarElement1, this.monthCalendar1 );

    BaseElement dataGridElement1 = htmelements["dataGrid1"] as BaseElement;
    new CustomControlBase( dataGridElement1, this.dataGrid1 );
}
```

5. Add a handler for the **Load** event and load the HTML document resource into the **HTMLUI** control.

[C#]

```
this.Load += new System.EventHandler(this.Form1_Load);

private void Form1_Load(object sender, System.EventArgs e)
{
    LoadHTMLResource();
}

private bool LoadHTMLResource()
{
    bool success = false;
    try
    {
        // Gets the Assembly that contains the code that is currently executing
        _assembly = Assembly.GetExecutingAssembly();
    }
}
```

```
// Loads the specified manifest resource from the Assembly
_htmlStream =
(Stream)_assembly.GetManifestResourceStream("HTMLUICustomControls.customcontrol
s.htm");
if(_htmlStream != null)
{
this.htmluiControl1.LoadHTML(_htmlStream);
success = true;
}
}
catch(Exception ex)
{
MessageBox.Show(ex.ToString());
}
return success;
}
```

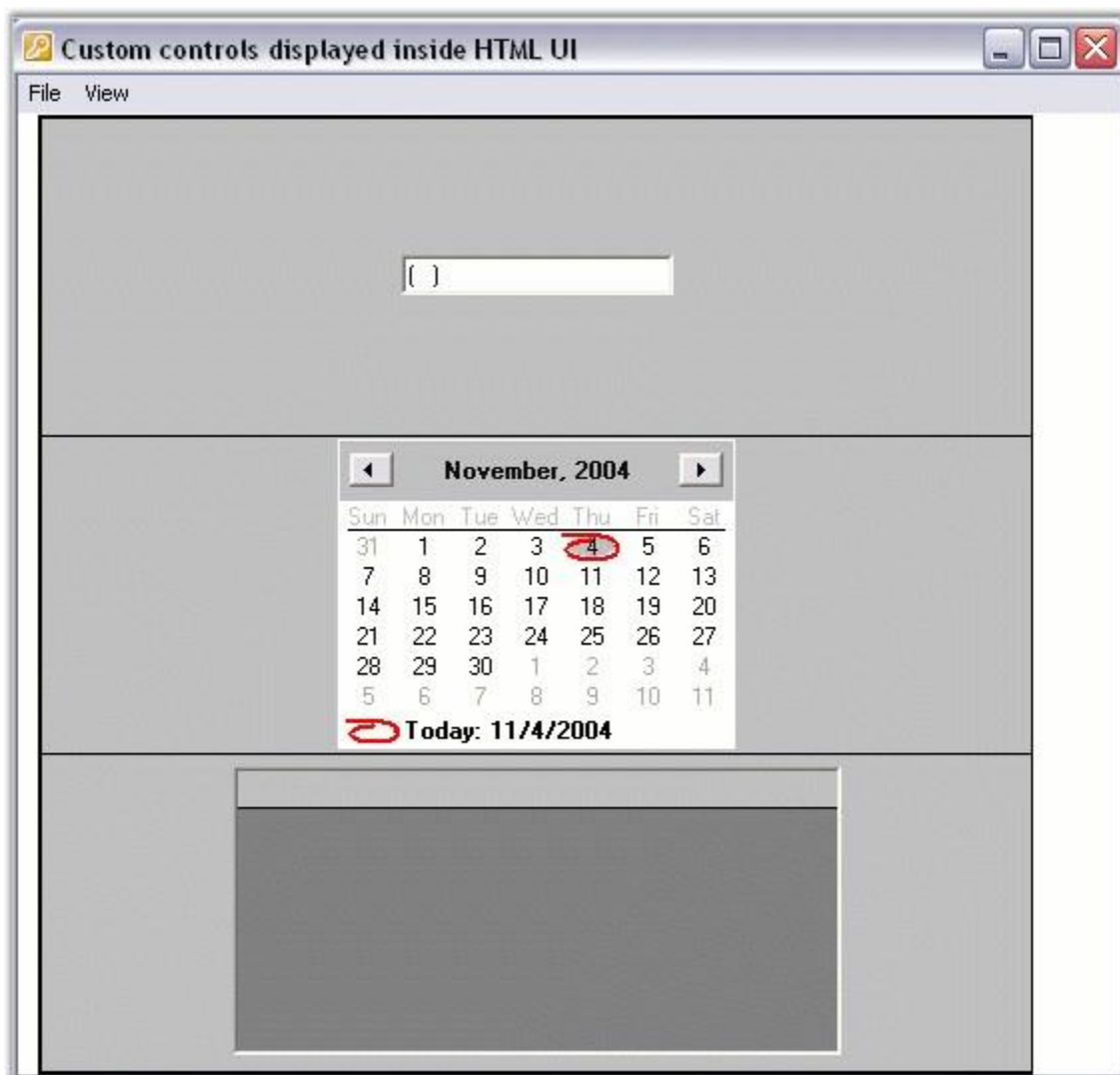


Figure 11: Document Loaded into HTMLUI Control

4 Concepts And Features

The HTMLUIControl is the main control of the HTMLUI library. The control exposes several properties, methods and events to load, display and interact with rich HTML-based user interfaces.

Creating the HTMLUI Control

The HTMLUI control can be created by dragging it from the Visual Studio .NET toolbox, just like any other Windows Forms control.

The following code snippet illustrates how to create a HTMLUIControl programmatically.

```
[C#]

// Initialize a HTMLUIControl.
this.htmluiControl1 = new Syncfusion.Windows.Forms.HTMLUI.HTMLUIControl();
```

Important Properties

The following properties help you to get started with the HTMLUIControl.

Property	Description
StartupDocument	This is the path to the HTML document that will be loaded when the HTMLUIControl is called. Using this property to set a document is the simplest means to load an HTML document.
Text	The HTMLUIControl does not display the Text property as text. This is the HTML that will be rendered as in the HTMLUIControl. This is the equivalent of the View Source option in a traditional web browser.
Document	This property provides access to all the display HTML elements programmatically.

Important Methods

The following methods help you to get started with the HTMLUIControl.

Method	Description
LoadHTML	This method is used to load the HTML document.
LoadCSS	This method is used to load CSS styles from file and refresh current

	document.
--	-----------

4.1 Loading HTML

HTML documents available at various resources can be easily loaded into the HTMLUI control. Some of the resources from where the HTML documents can be loaded are as follows.

- HTML files available on the disk
- HTML files available in the embedded resource
- As links from one HTML document to the other
- HTML files available in the URI
- HTML which is in the form of a text

An HTML document can be loaded into the HTMLUI control in two ways.

4.1.1 Loading As a Startup Document

There may be situations where the HTML document is to be loaded initially at startup. An HTML document is loaded at startup for the front page applications. It may be an introductory page or a page that contains information regarding forthcoming pages. An HTML document can be loaded at Startup by two ways:

- Using the Properties window
- Coding

Using the Properties window involves specifying the location of the Startup HTML file in the **StartupDocument** property available within the properties window for the HTMLUI control or by clicking the link **Load from file** shown at the bottom of the properties window.

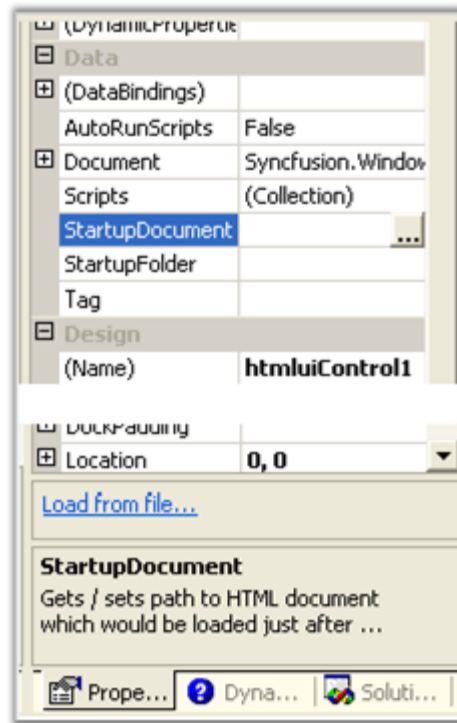


Figure 12: HTMLUI Properties Grid

While coding for the Startup Document, it should be written in the **form_load** event that is handled before the form is displayed for the first time.

[C#]

```
// Get or Set the path to the Startup Document for the control.
private void Form1_Load(object sender, System.EventArgs e)
{
    this.htmluiControl1.StartupDocument = @"C:\MyProjects\Startup\startup_page.htm";
}
```

[VB.NET]

```
' Get/Set the path to the Startup Document for the control.
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Me.htmluiControl1.StartupDocument = "C:\MyProjects\Startup\startup_page.htm"
End Sub
```

The following image illustrates Loading an HTML document as the Startup Document.

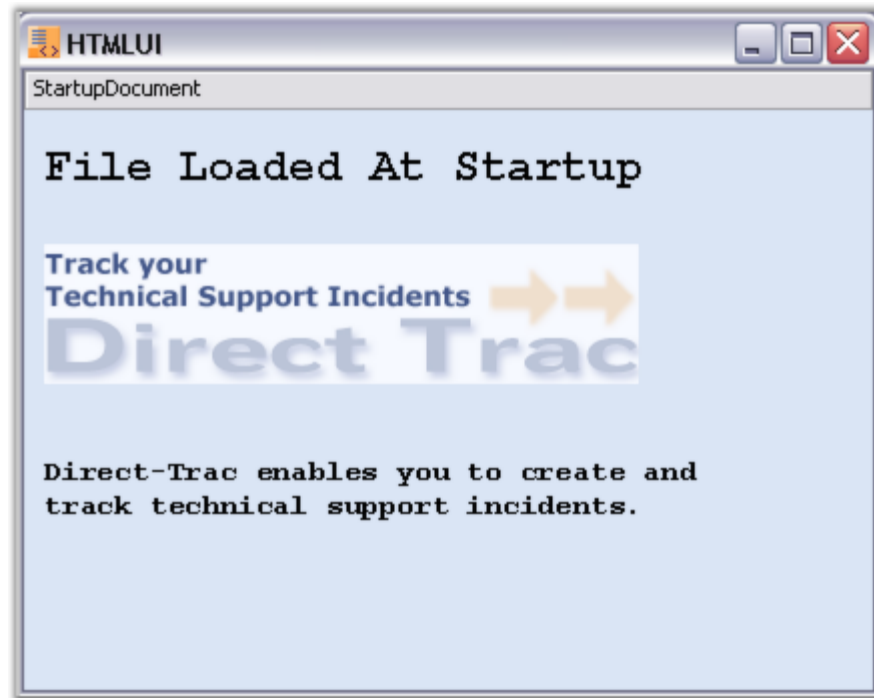


Figure 13: Loading an HTML document as the Startup Document

4.1.1.1 Startup File Sample

This sample demonstrates the implementation of Startup Document by using HTML file in HTMLUI.

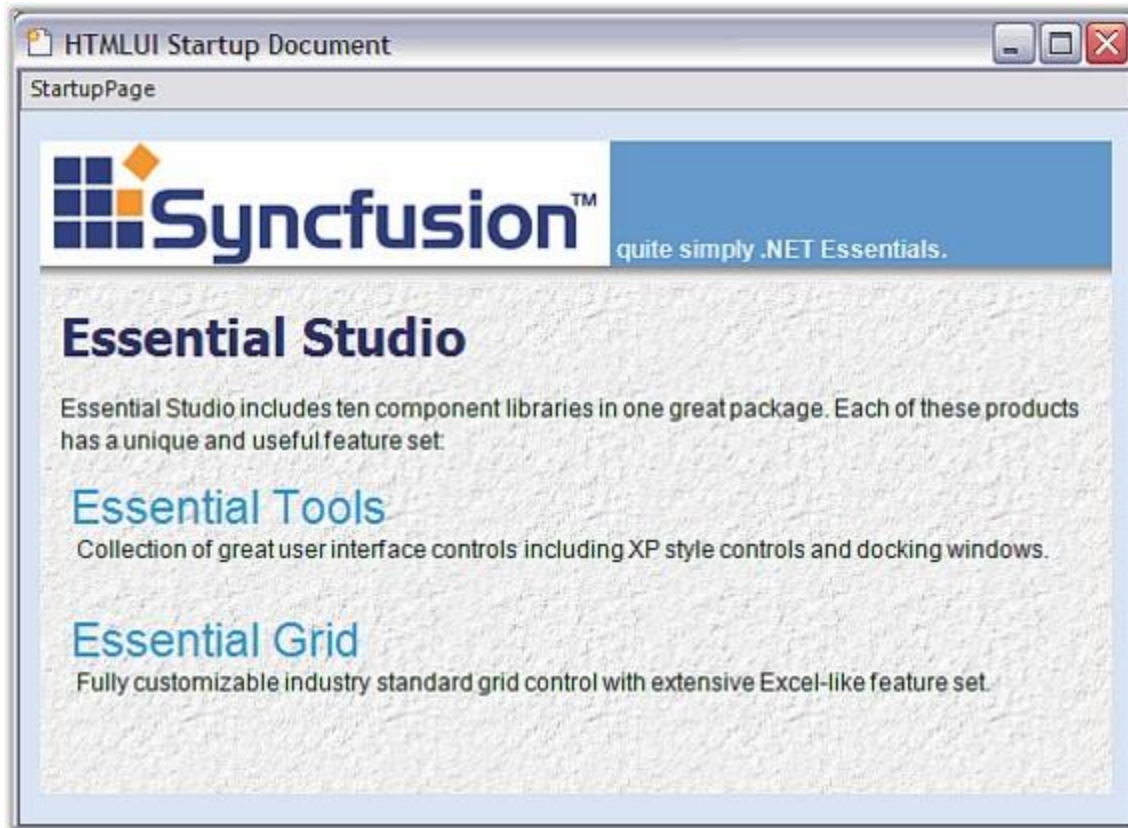


Figure 14: Implementation of Startup Document by using HTML File

By default, this sample can be found under the following location:

C:\Documents and Settings\<username>\My Documents\Syncfusion\Essential Studio\<Version Number>\Windows\HTMLUI.Windows\Samples\2.0>Loading HTML\HTMLUIStartupDocument

4.1.2 Loading At Run Time

HTML documents can also be loaded during runtime; for example, in a file link where an HTML file may link to another file. In that case a new file is loaded in the control after the one that was initially loaded.

The various ways of loading the document during the runtime from various resources are as follows:

- Loading the file from disk
- As links from one HTML document to the other
- Loading the file from URI

- Loading HTML in the form of text
- Loading the file from Resource

The following sections explains the above concepts in detail.

4.1.2.1 Loading the File From Disk

The HTML file that is located in the user's disk can be loaded into the HTMLUIControl. It is loaded by specifying the location of the file in the disk.

[C#]

```
// Load the specified HTML Document from user's drive.  
string filepath = @"C:\MyProjects\LoadHTML\FromDisk.htm";  
this.htmluiControl1.LoadHTML(filepath);
```

[VB.NET]

```
'Load the specified HTML Document from user's drive.  
Private filepath As String = "C:\MyProjects\LoadHTML\FromDisk.htm"  
Me.HtmluiControl1.LoadHTML(filepath)
```

The following image shows file loaded from the User's Drive.

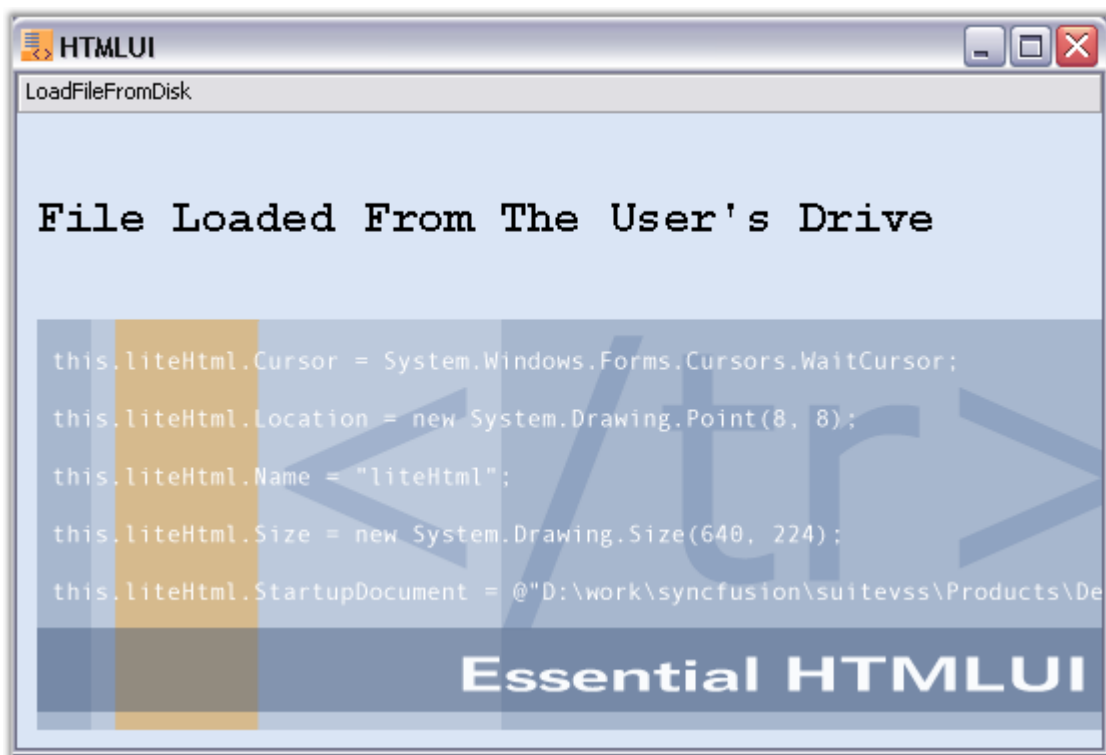


Figure 15: Loading HTML document from the User's Drive into the HTMLUI Control

4.1.2.1.1 Load File From Disk Sample

This sample demonstrates the implementation of Loading a file from Disk by using HTMLUI.

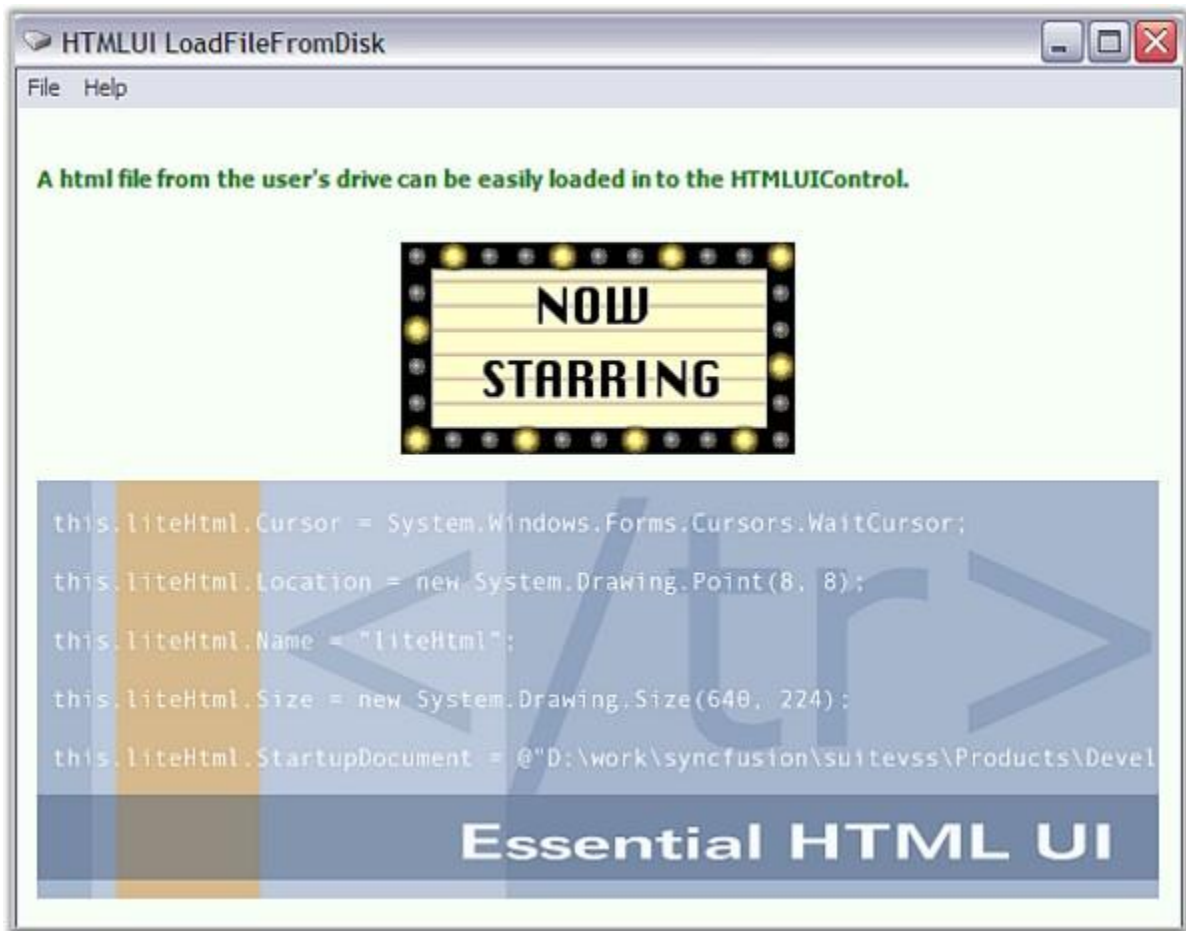


Figure 16: LoadFileFromDisk Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0>Loading HTML\HTMLUILoadFileFromDisk

4.1.2.2 As Links From One HTML Document To Another

The HTMLUI supports **Link** property. Links in HTML code are easily invoked in HTMLUI Control. The main document that contains links to other documents is loaded into the HTMLUI control. The linked document is loaded by clicking the respective link in the main document.

[C#]

```
// Load the specified HTML Document that contains link for another document.
string filepath = @"C:\MyProjects\LoadHTML\Main.htm";
this.htmluiControl1.LoadHTML(filepath);
```

[VB.NET]

```
' Load the specified HTML Document that contains link for another document.
Private filepath As String = "C:\MyProjects\LoadHTML\Main.htm"
Me.HtmluiControl1.LoadHTML(filepath)
```

HTML Code

An HTML document containing file links is illustrated by the code given below:

[HTML]

```
<HTML>
<HEAD>
<title>FILE LINK</title>
</HEAD>
<body bgColor="#ffffff">
<P>THIS FORM IS A SAMPLE TO SHOW FILE LINKS</P>
<FONT color="#66ffff" size="6">
<A href="MODEL1.htm">To link1</a>
</FONT>
<P>
<FONT color="#66ffff" size="6">
<A href="MODEL2.htm">To link2</a>
</FONT>
</P>
</body>
</HTML>
```

The following image shows file links that link to another HTML Document.

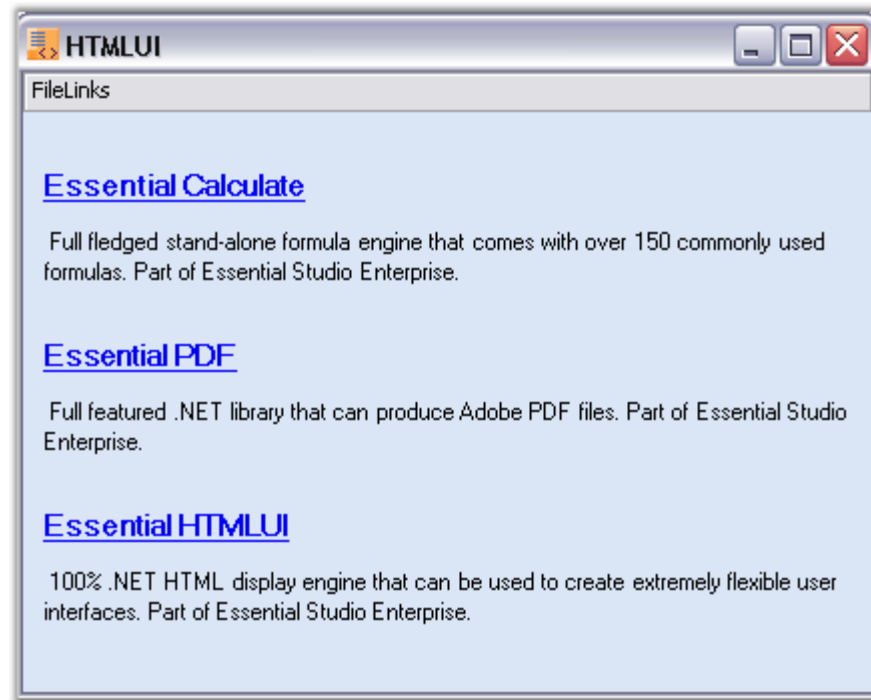


Figure 17: Loading Linked documents in the HTMLUI Control

4.1.2.2.1 File Links Sample

This sample demonstrates how HTML files can be linked from one document to another by using HTMLUI.



Figure 18: File Links Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0>Loading HTML\File Links

4.1.2.3 Loading the File From URI

HTML contents can also be loaded from the URI (Uniform Resource Identifier). This is a great advantage of HTMLUI that it can be used for browsing purposes like popular web browsers.

[C#]

```

// Load the HTML document from the specified Uri in to HTMLUI Control.
Uri uri = new Uri("http://www.syncfusion.com");
htmluiControl1.LoadHTML(uri);
    
```

[VB.NET]

```

' Load the HTML document from the specified Uri in to HTMLUI Control.
    
```



```
Private uri As Uri = New Uri("http://www.syncfusion.com")
HtmluiControl1.LoadHTML(uri)
```

A new URI has to be declared in the code with the path from which the URI has to be loaded, as shown in the above example. The **Uri** class provides an object representation of a URI and also provides easy access to the parts of the URI.

4.1.2.4 Loading HTML Which Is In the Form Of Text

The HTML code sometimes can be directly written and stored as a string. The HTML code available in the form of string is loaded into the HTMLUI Control by using the **LoadFromString** method and the HTML contents will be displayed in the HTMLUI control.

[C#]

```
// Load HTML Document from String.
string htmlCode = "<HTML>
<HEAD>
<TITLE>HI</TITLE>
</HEAD>
<BODY bgcolor='#ffffff'>
<INPUT type='button' id='btn' /></INPUT>
</BODY>
</HTML>";
this.htmluiControl1.LoadFromString(htmlCode);
```

[VB.NET]

```
' Load HTML Document from String
Private htmlCode As String = "<HTML>
<HEAD>
<TITLE>HI</TITLE>
</HEAD>
<BODY bgcolor='#ffffff'>
<INPUT type='button' id='btn' /></INPUT>
</BODY>
</HTML>"
Me.HtmluiControl1.LoadFromString(htmlCode)
```

4.1.2.5 Load the File from Resource

The HTML file can be loaded as an Embedded Resource in the HTMLUI control. The procedure to be followed for making an HTML file as an embedded resource is discussed below.

1. Open the **Solution Explorer** from the **View** menu of the Menu Bar.

2. Right-click on the **C#** file name in the **Solution Explorer**. A menu opens.
3. Click the **Add** tab; a sub-menu is displayed.

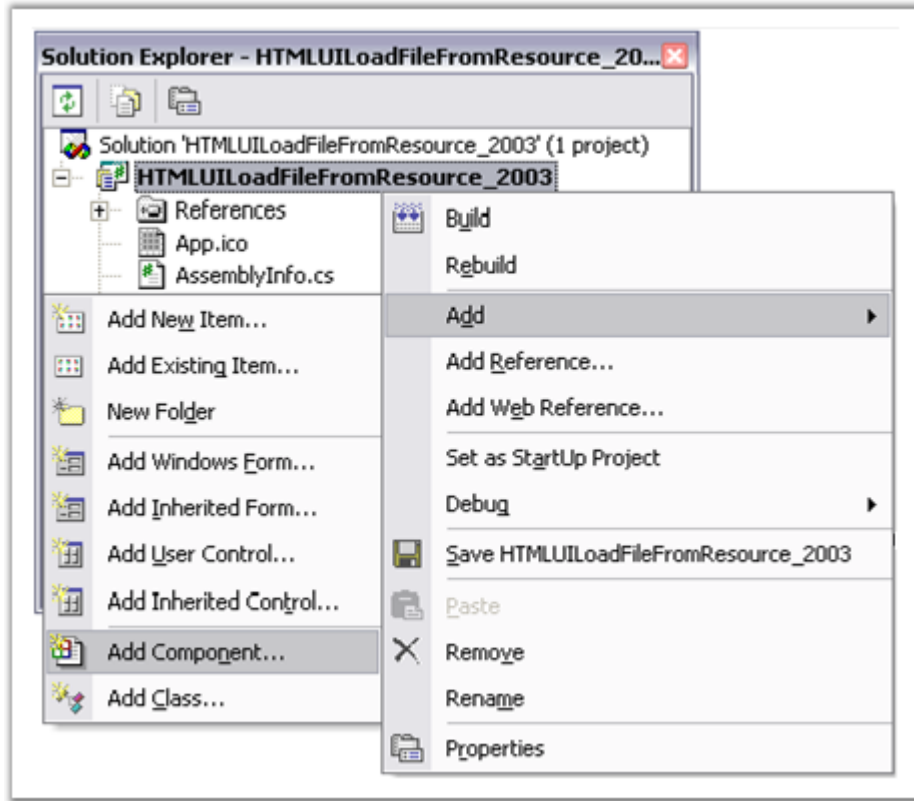


Figure 19: Menu options in adding a new HTML File

4. In the sub-menu, click **AddNewItem** ; a template wizard is displayed.
5. In the wizard, select HTML Page. The default name for the page is 'HTMLPage1.htm'.
6. You can change the name by using the **Name** tab given at the bottom of the wizard.

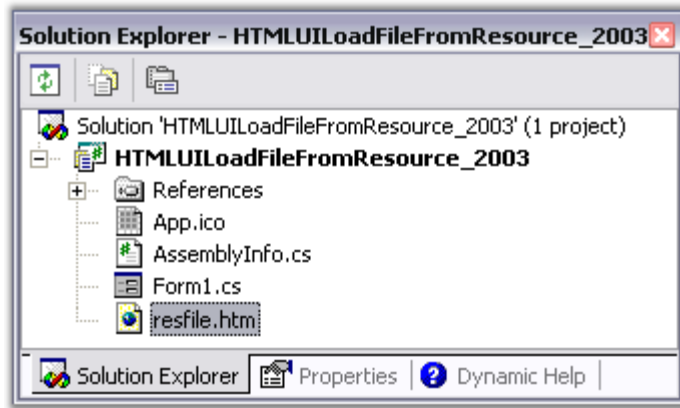


Figure 20: Tree view of the Solution Explorer

7. The HTML file will be shown in the **Solution Explorer** as shown in the figure above.
8. In the properties grid of the resource HTML file, specify its **BuildAction** as the Embedded Resource.

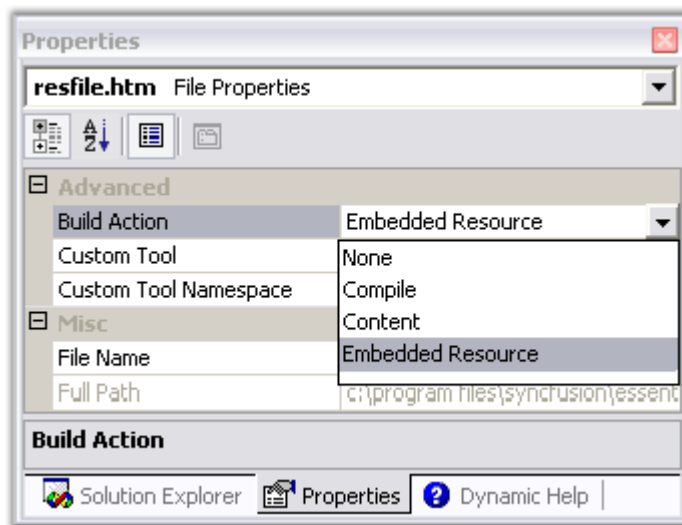


Figure 21: Properties Window of the Resource HTML File

The file can be retrieved from the resource by using the following C# code.

[C#]

```
// Load the specified HTML file which is marked as the project's embedded resource.  
htmlStream = (Stream)Assembly.GetExecutingAssembly().GetManifestResourceStream
```

```
("LoadingFileFromResource.resfile.htm");  
  
this.htmluiControl1.LoadHTML(htmlStream);
```

[VB.NET]

```
' Load the specified HTML file which is marked as the project's embedded resource.  
Private htmlStream = CType(System.Reflection.Assembly.GetExecutingAssembly().  
GetManifestResourceStream ("LoadingFileFromResource.resfile.htm"), Stream)  
  
Me.HtmluiControl1.LoadHTML(htmlStream)
```

It is necessary to invoke the **System.IO** and **System.Reflection** namespaces to use the classes and their methods used in the code above.

The **System.Reflection.Assembly.GetExecutingAssembly** method gets the assembly from which the code is currently running from and the **GetManifestResourceStream** method of the same class loads the specified manifest resource from the assembly.

The **System.IO.Stream** is used to provide a generic view of sequence of bytes when the IO in the assembly is referred.

Note: The string entered inside the *GetManifestResourceStream* method is in reference to the Default namespace found in the Properties window of the C# file in the Solution Explorer. This may vary for the users.

The following image shows file loaded from an embedded resource.



Figure 22: Loading HTML File from an Embedded Resource into the HTMLUI Control

4.1.2.5.1 Load Resource File Sample

This sample demonstrates the implementation of Loading Embedded Resource Files by using HTMLUI.

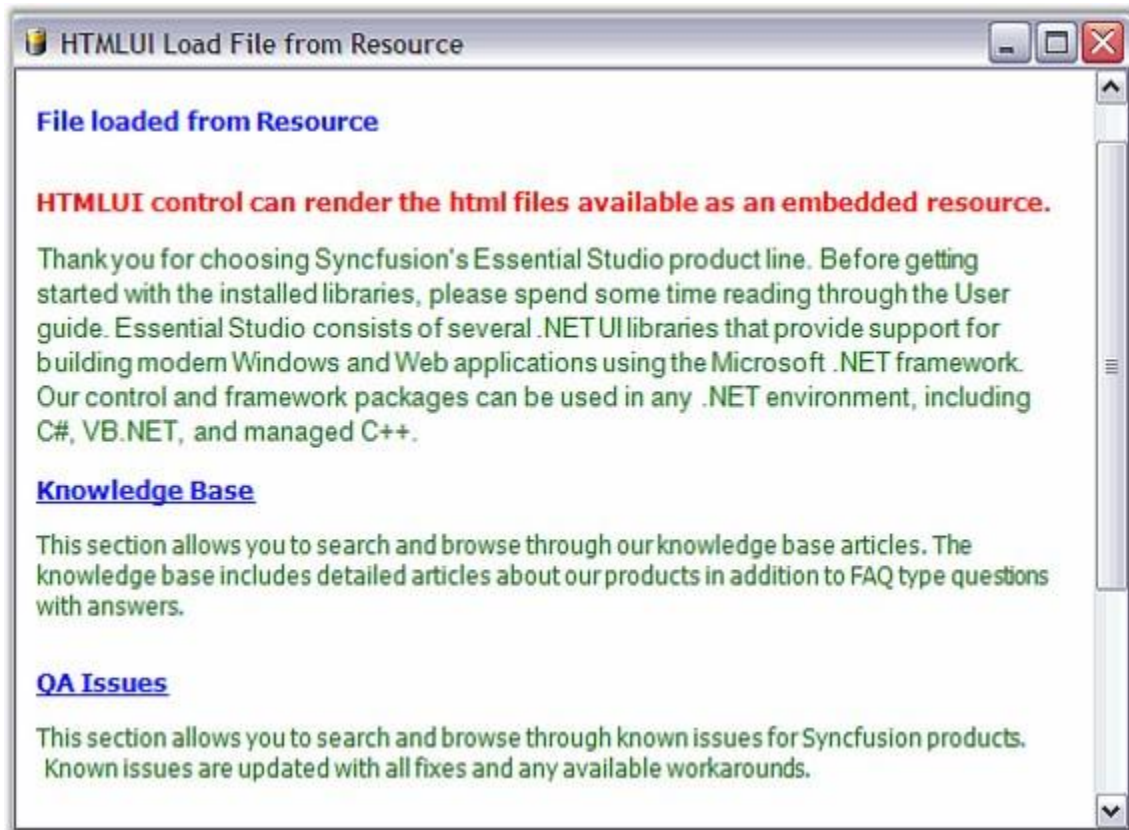


Figure 23: HTMLUILoadFileFromResource Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0>Loading HTML\HTMLUILoadFileFromResource

4.2 MHT Formats

MHTML enables you to send and receive Web pages and other HTML documents by using e-mail programs. MHTML enables you to embed images directly into the body of your e-mail messages rather than attaching them to the message. MHTML uses MIME, which provides facilities to allow multiple objects in a single Internet e-mail message {comma removed} to represent formatted multifont text messages, non-textual materials such as images, and so on.

HTMLUI supports the usage of the simple MHTML files. The HTMLUI control allows the user to load the MHTML files from the user's drive with the help of the **LoadHTML** method.

[C#]

```
// LoadHTML() method for loading the mht files in HTMLUI is limited to use
filenames only
// No overloads are supported.
this.htmluiControl.LoadHTML(@"C:\MyProjects\MHTSupport\sample.mht");
```

[VB.NET]

```
' LoadHTML() method for loading the mht files in HTMLUI is limited to use
filenames only
' No overloads are supported.
Me.htmluiControl.LoadHTML("C:\MyProjects\MHTSupport\sample.mht")
```

4.3 Control Events

HTMLUI control comes with a rich set of events to help the application developer in keeping track of the execution. These events are programmed based on the Event arguments containing data related to the event.

The events executed by the HTMLUI control are as follows:

- [LinkClicked Event](#)
- [LoadStarted Event](#)
- [LoadFinished Event](#)
- [LoadError Event](#)
- [PreRenderDocument Event](#)
- [ShowTitleChanged Event](#)
- [TitleChanged Event](#)

LinkClicked Event

This event is raised after the hyperlink is clicked and before the hyperlink tries to load a new resource. The event properties associated with the Link Forward Event Arguments are as follows.

- **Cancel:** A boolean value which indicates whether the default processing of resource loading should be canceled or not
- **Path:** Specifies the location of the resource

[C#]

```
// Event that is to be raised after the hyperlink was clicked and before the
// hyperlink tries to load
// a new resource.
this.htmluiControl1.LinkClicked += new
Syncfusion.Windows.Forms.HTMLUI.LinkForwardEventHandler(this.htmluiControl1_LinkClick
ed);

private void htmluiControl1_LinkClicked(object sender,
Syncfusion.Windows.Forms.HTMLUI.LinkForwardEventArgs e)
{
    e.Cancel = true;
    Form2 form2 = new Form2(GetFilesLocation() + e.Path);
    form2.Show();
}
```

[VB.NET]

```
' Event that is to be raised after the hyperlink was clicked and before the hyperlink
tries to load
' a new resource.
Me.HtmluiControl1.LinkClicked += New
Syncfusion.Windows.Forms.HTMLUI.LinkForwardEventHandler(Me.htmluiControl1_LinkClicked
)

Private Sub htmluiControl1_LinkClicked(ByVal sender As Object, ByVal e As
Syncfusion.Windows.Forms.HTMLUI.LinkForwardEventArgs)
    e.Cancel = True
    Dim form2 As Form2 = New Form2(GetFilesLocation() + e.Path)
    form2.Show()
End Sub
```

LoadStarted Event

This event is raised when a new HTML document has started loading into the HTMLUI control from the specified resource.

[C#]

```
// Event that is to be raised when the HTMLUI control starts loading a new html
document.
this.htmluiControl1.LoadStarted += new
System.EventHandler(this.htmluiControl1_LoadStarted);

private void htmluiControl1_LoadStarted(object sender, System.EventArgs e)
{
    Console.WriteLine("Started Loading...");
}
```

[VB.NET]

```
' Event that is to be raised when the HTMLUI control starts loading a new html
document.
Me.HtmluiControl1.LoadStarted += New
System.EventHandler(Me.htmluiControl1_LoadStarted)

Private Sub htmluiControl1_LoadStarted(ByVal sender As Object, ByVal e As
System.EventArgs)
    Console.WriteLine("Started Loading...")
End Sub
```

LoadFinished Event

This event is raised after the loading of HTML document inside the HTMLUI control is completed.

[C#]

```
// Event that is to raised after the HTML document have been rendered in the control.
this.htmluiControl1.LoadFinished += new
System.EventHandler(this.htmluiControl1_LoadFinished);

private void htmluiControl1_LoadFinished(object sender, System.EventArgs e)
{
    Console.WriteLine("Load successfully completed");
}
```

[VB.NET]

```
' Event that is to raised after the HTML document have been rendered in the control.
Me.HtmluiControl1.LoadFinished += New
System.EventHandler(Me.htmluiControl1_LoadFinished)

Private Sub htmluiControl1_LoadFinished(ByVal sender As Object, ByVal e As
System.EventArgs)
    Console.WriteLine("Load successfully completed")
End Sub
```

LoadError Event

This event is raised when an error occurs during loading or rendering an HTML document from the specified resource. The LoadEventArgs contains the following property that defines the data related to the action of this event.

- **Document:** Specifies the document whose rendering triggers the execution of the LoadError event

[C#]

```
// Event that is to be raised when an error occurs during HTML document is being
rendered in the HTMLUI control.
this.htmluiControl1.LoadError += new
Syncfusion.Windows.Forms.HTMLUI.LoadErrorHandler(this.htmluiControl1_LoadError);

private void htmluiControl1_LoadError(object sender,
Syncfusion.Windows.Forms.HTMLUI.LoadEventArgs e)
{
    Console.WriteLine("Error loading due to"+ e.ToString());
}
```

[VB.NET]

```
' Event that is to be raised when an error occurs during HTML document is being
rendered in the HTMLUI control.
```

```
Me.HtmluiControll1.LoadError += New
Syncfusion.Windows.Forms.HTMLUI.LoadErrorHandler(Me.htmluiControll1_LoadError)

Private Sub htmluiControll1_LoadError(ByVal sender As Object, ByVal e As
Syncfusion.Windows.Forms.HTMLUI.LoadEventArgs)
    Console.WriteLine("Error loading due to" + e.ToString())
End Sub
```

PreRenderDocument Event

This event is raised when the elements in the HTML document are created in the HTMLUI control, but their size and location are not calculated yet.

[C#]

```
// Event that is to be raised when a tree of element has been created and their size
and location have not been calculated yet.
this.htmluiControll1.PreRenderDocument += new
Syncfusion.Windows.Forms.HTMLUI.PreRenderDocumentEventHandler
(this.htmluiControll1_PreRenderDocument);

private void htmluiControll1_PreRenderDocument(object sender,
Syncfusion.Windows.Forms.HTMLUI.PreRenderDocumentArgs e)
{
    Console.WriteLine("This is the Prerender document event");
}
```

[VB.NET]

```
' Event that is to be raised when a tree of element has been created and their size
and location have not been calculated yet.
Me.HtmluiControll1.PreRenderDocument += New
Syncfusion.Windows.Forms.HTMLUI.PreRenderDocumentEventHandler
(Me.htmluiControll1_PreRenderDocument)

Private Sub htmluiControll1_PreRenderDocument(ByVal sender As Object, ByVal e As
Syncfusion.Windows.Forms.HTMLUI.PreRenderDocumentArgs)
    Console.WriteLine("This is the Prerender document event")
End Sub
```

ShowTitleChanged Event

This event is raised after the **ShowTitle** property of the HTMLUI control is changed. The event handler receives its data from the ValueChangedEventArgs. The following properties are associated with the ShowTitleChanged event handling.

- **Empty**-Gets the instance of the class that is found to be empty or having null value

- **newValue**-Indicates the current value of the ShowTitle property
- **oldValue**-Indicates the old value of the ShowTitle property

[C#]

```
// Event that is raised after the ShowTitle property of the HTMLUI control is changed.
this.htmluiControl1.ShowTitleChanged += new
Syncfusion.Windows.Forms.HTMLUI.ValueChangedEventHandler(this.htmluiControl1_ShowTitleC
eChanged);

private void htmluiControl1_ShowTitleChanged(object sender, ValueChangedEventArgs e)
{
    MessageBox.Show("ShowTitle Changed");
}
```

[VB.NET]

```
' Event that is raised after the ShowTitle property of the HTMLUI control is changed.
Me.HtmluiControl1.ShowTitleChanged += New
Syncfusion.Windows.Forms.HTMLUI.ValueChangedEventHandler(Me.htmluiControl1_ShowTitleC
hanged)

Private Sub htmluiControl1_ShowTitleChanged(ByVal sender As Object, ByVal e As
ValueChangedEventArgs)
    MessageBox.Show("ShowTitle Changed")
End Sub
```

TitleChanged Event

The TitleChanged event is raised after the **Title** property of the HTMLUI control is changed. The Title value can be set explicitly by the user or it can be extracted from the title tag of the HTML document that is to be loaded into the HTMLUI control.

The event handler receives its data from the ValueChangedEventArgs. The following properties are associated with the TitleChanged event handling.

- **newValue**: Gets the new value for the Title
- **oldValue**: Gets the old value that has been changed

[C#]

```
// Event is raised after the Title property of the HTMLUI control is changed.
this.htmluiControl1.TitleChanged += new
Syncfusion.Windows.Forms.HTMLUI.ValueChangedEventHandler
(this.htmluiControl1_TitleChanged);
private void htmluiControl1_TitleChanged(object sender, ValueChangedEventArgs e)
{
```

```
MessageBox.Show("Title Changed");
}
```

[VB.NET]

```
'Event is raised after the Title property of the HTMLUI control is changed.
Me.HtmluiControl1.TitleChanged += New
Syncfusion.Windows.Forms.HTMLUI.ValueChangedEventHandler(Me.htmluiControl1_TitleChang
ed)

Private Sub htmluiControl1_TitleChanged(ByVal sender As Object, ByVal e As
ValueChangedEventArgs)
    MessageBox.Show("Title Changed")
End Sub
```

4.3.1 HTMLUI Control Events Sample

This sample illustrates the different events executed by the HTMLUI control.

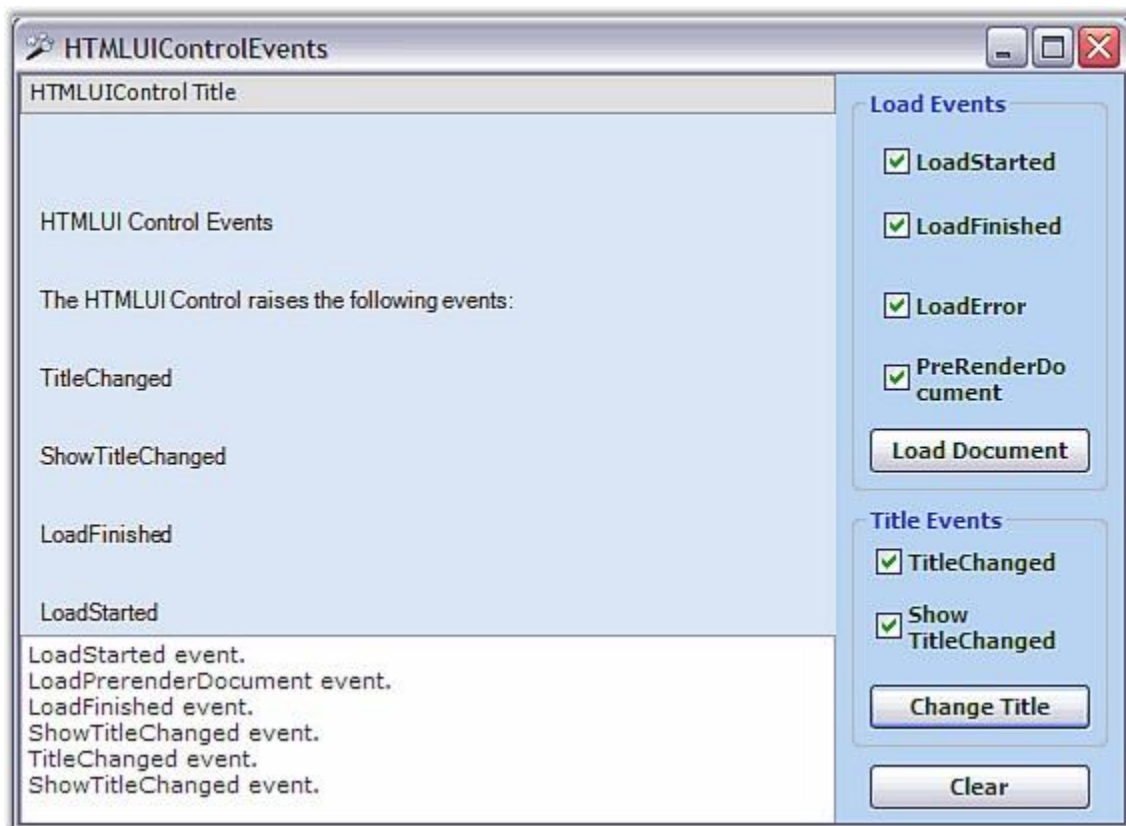


Figure 24: HTMLUIControlEvents Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\ Samples\2.0\HTMLUI Events\HTMLUIControlEvents

4.4 Element Events

Each HTML element in an HTML document is made to support events, such as **Click**, **DoubleClick**, **MouseMove**, **KeyPress**, and so on just like the Windows forms controls.

[HTML]

```
<html>
  <body>
    <input type="text" id="text1"/>
  </body>
</html>
```

[C#]

```
// Object declaration for the textarea element in the html document rendered in the
// control.
private void htmluiControl1_LoadFinished(object sender, System.EventArgs e)
{
    Hashtable htmlElements = this.htmluiControl1.Document.GetElementsByUserIdHash();
    BaseElement textElement = htmlElements["text1"] as BaseElement;

    // Event handlers declaration for the events on the html elements.
    textElement.Click += new EventHandler( textElement_Click );
    textElement.KeyDown += new EventHandler( textElement_KeyDown );
    textElement.MouseEnter += new EventHandler( textElement_MouseEnter );
}

// HTML element Click event definition.
private void textElement_Click( object sender, EventArgs e )
{
    Console.WriteLine("Click Event Handled");
}

// HTML element KeyDown event definition.
private void textElement_KeyDown( object sender, EventArgs e )
{
    Console.WriteLine("KeyDown Event Handled");
}
```

```
// HTML element MouseEnter event definition.
private void textElement_MouseEnter( object sender, EventArgs e )
{
    Console.WriteLine("MouseDown Event Handled");
}
```

[VB.NET]

```
' Object declaration for the textarea element in the html document rendered in the
control.
Private Sub htmluiControl1_LoadFinished(ByVal sender As Object, ByVal e As
System.EventArgs)
Dim htmlElements As Hashtable = Me.htmluiControl1.Document.GetElementsByUserIdHash()
Dim textElement As BaseElement = CType(IIf(.TypeOf htmlElements("text1") Is
BaseElement, htmlElements("text1"), Nothing), BaseElement)

' Event handlers declaration for the events on the html elements.
AddHandler textElement.Click, AddressOf textElement_Click
AddHandler textElement.KeyDown, AddressOf textElement_KeyDown
AddHandler textElement.MouseEnter, AddressOf textElement_MouseEnter
End Sub

' HTML element Click event definition.
Private Sub textElement_Click(ByVal sender As Object, ByVal e As EventArgs)
Console.WriteLine("Click Event Handled")
End Sub

' HTML element KeyDown event definition.
Private Sub textElement_KeyDown(ByVal sender As Object, ByVal e As EventArgs)
Console.WriteLine("KeyDown Event Handled")
End Sub

' HTML element MouseEnter event definition.
Private Sub textElement_MouseEnter(ByVal sender As Object, ByVal e As EventArgs)
Console.WriteLine("MouseDown Event Handled")
End Sub
```

Another important feature of the HTMLUI is its **Bubbling Event** architecture. With this architecture, a single common event handler defined for a particular event of the parent can be used for all the Child Elements bound to that parent while executing the same event.

[HTML]

```
<html>
  <body>
    <input type="button" id="button1"/>
    <br/>
    <input type="button" id="button2"/>
  </body>
```

```
</html>
```

[C#]

```
private void htmluiControl1_LoadFinished(object sender, System.EventArgs e)
{
    IHTMLElement[] htmlelement = this.htmluiControl1.Document.GetElementsByName("body");
    htmlelement[0].MouseLeave += new EventHandler(body_MouseLeave);
}

// Event occurs when the mouse pointer leaves the control.
private void body_MouseLeave(object sender, EventArgs e)
{
    // Converts the EventArgs object to BubblingEventArgs type if possible.
    BubblingEventArgs bargs = HTMLUIControl.GetBublingEventArgs(e);

    // Returns the first sender of the event.
    BaseElement elem = bargs.RootSender as BaseElement;

    if( elem != null && elem is INPUTElementImpl)
    {
        if(elem.ID == "button1")
        {
            this.label1.Text = "Mouse just left Button 1";
        }
        else if(elem.ID == "button2")
        {
            this.label1.Text = "Mouse just left Button 2";
        }
    }
}
```

[VB.NET]

```
' Event occurs when the mouse pointer leaves the control.
Private Sub htmluiControl1_LoadFinished(ByVal sender As Object, ByVal e As
System.EventArgs)
    Dim htmlelement As IHTMLElement() =
    Me.htmluiControl1.Document.GetElementsByName("body")
    AddHandler htmlelement(0).MouseLeave, AddressOf body_MouseLeave
End Sub

Private Sub body_MouseLeave(ByVal sender As Object, ByVal e As EventArgs)

' Converts the EventArgs object to BubblingEventArgs type if possible.
Dim bargs As BubblingEventArgs = HTMLUIControl.GetBublingEventArgs(e)

' Returns the first sender of the event.
Dim elem As BaseElement = CType(If(OfType(bargs.RootSender Is BaseElement,
bargs.RootSender, Nothing), BaseElement)
```

```
If Not elem Is Nothing AndAlso TypeOf elem Is INPUTElementImpl Then
    If elem.ID = "button1" Then
        Me.label1.Text = "Mouse just left Button 1"
    ElseIf elem.ID = "button2" Then
        Me.label1.Text = "Mouse just left Button 2"
    End If
End If
End Sub
```

4.4.1 HTMLUI Bubbling Events Sample

This sample demonstrates the implementation of Bubbling Event architecture in HTMLUI.

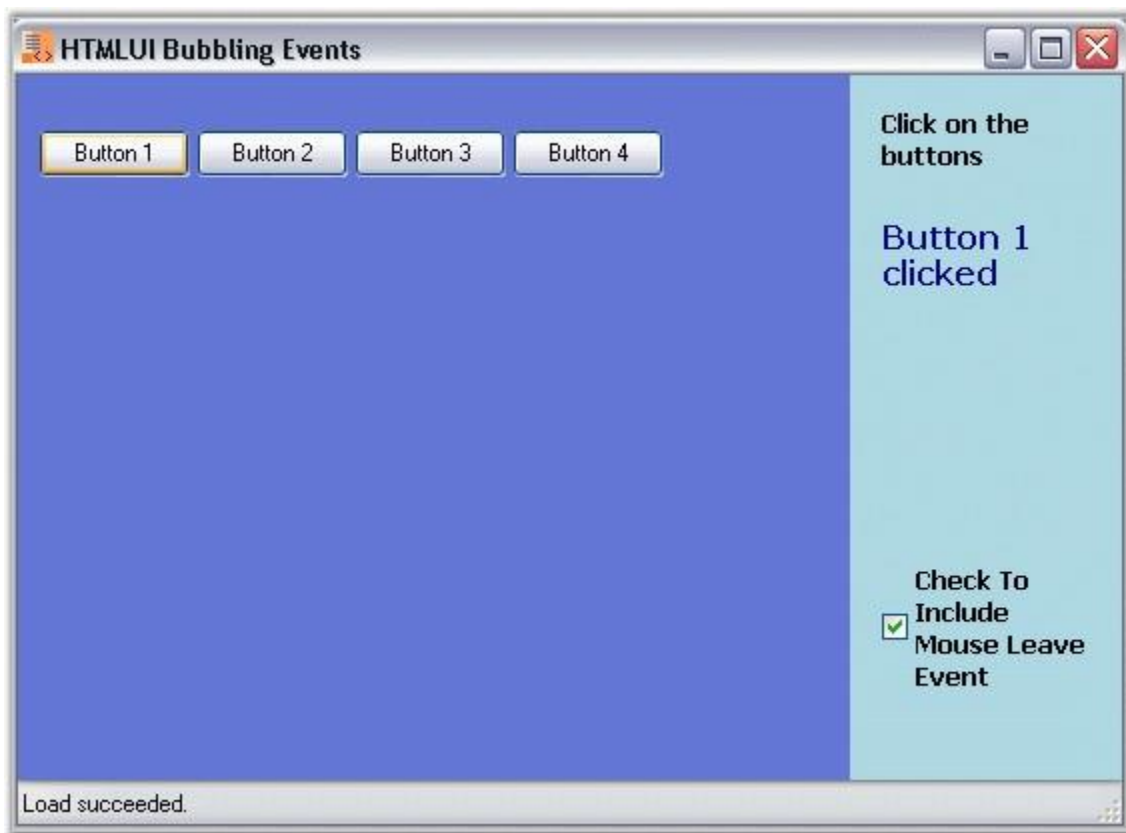


Figure 25: HTMLUIBubblingEvents Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0\HTMLUI Events\HTMLUIBubblingEvents

4.4.2 HTMLUI Element Events Sample

This sample shows how element events are handled for creating effective user interfaces.

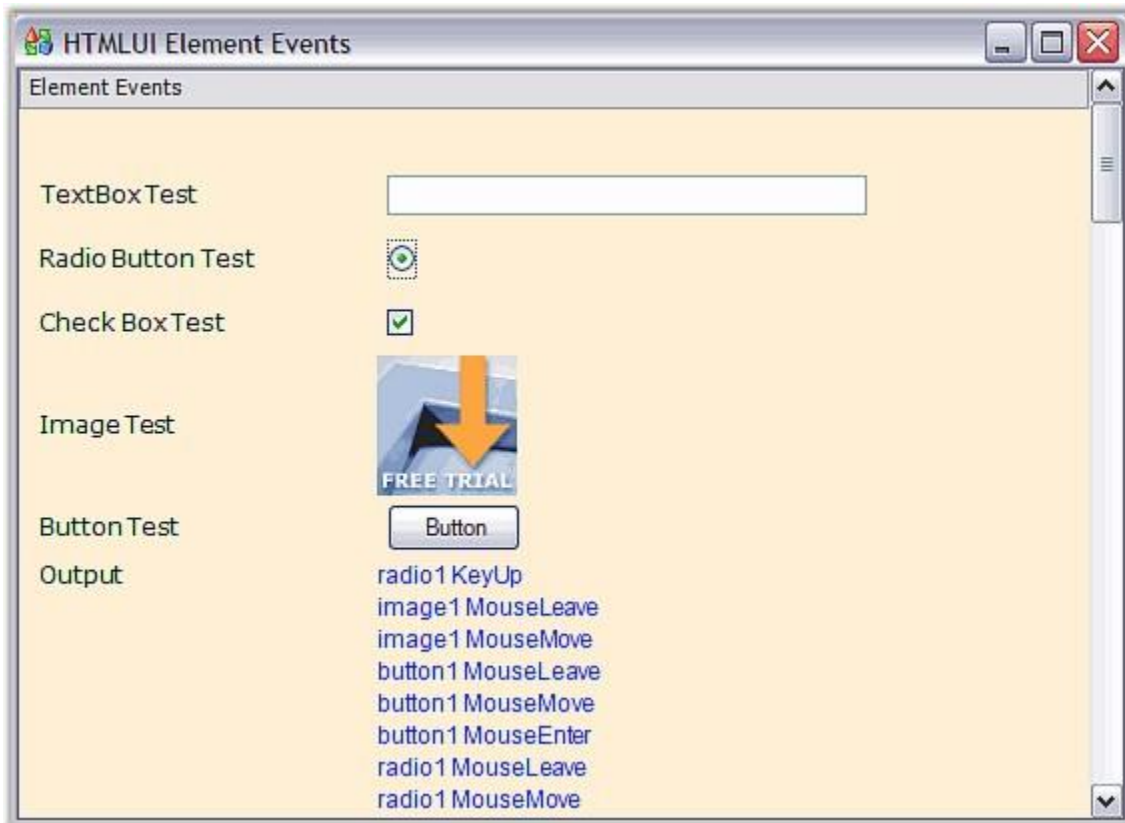


Figure 26: HTMLUIElementEvents Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\ Samples\2.0\HTMLUI Events\HTMLUIElementEvents

4.5 HTML Elements

HTMLUI supports various elements in an HTML document for rendering and presenting them to the user and also allows the user to dynamically access the elements to produce rich, customized user interfaces. Each HTML element defines properties and methods which can be used for customization.

The property **SupportedEvents** and the method **MergeSupportedEvents** are common to most HTML elements.

SupportedEvents

This property returns an array of events supporting the element.

[C#]

```
// SupportedEvents property returns an array of events supporting the element.
Hashtable htmlelements = this.htmluiControl1.Document.GetElementsByUserIdHash();
BRElementImpl br = htmlelements["br"] as BRElementImpl;
this.label1.Text = this.br.SupportedEvents.Length.ToString();
```

[VB.NET]

```
' SupportedEvents property returns an array of events supporting the element.
Private htmlelements As Hashtable =
Me.HtmluiControl1.Document.GetElementsByUserIdHash()
Private tr As BRElementImpl = CType(IIf(OfType htmlelements("br") Is BRElementImpl,
htmlelements("br"), Nothing), BRElementImpl)
Private Me.label1.Text = Me.br.SupportedEvents.Length.ToString()
```

MergeSupportedEvents

The **MergeSupportedEvents** method is used to merge the standard and special events.

[C#]

```
// MergeSupportedEvents method is to merge the standard and special events.
Hashtable htmlelements = this.htmluiControl1.Document.GetElementsByUserIdHash();
INPUTElementImpl txt = htmlelements["txt"] as INPUTElementImpl;
string[] specialEvents = new string[2];
specialEvents[0] = "Yes";
specialEvents[1] = "No";
MessageBox.Show("Before Merging:" + this.txt.SupportedEvents.Length.ToString());
this.txt.MergeSupportedEvents(specialEvents);
MessageBox.Show("After Merging:" + this.txt.SupportedEvents.Length.ToString());
```

[VB.NET]

```

' MergeSupportedEvents method is to merge the standard and special events.
Private htmlelements As Hashtable =
Me.HtmluiControll.Document.GetElementsByUserIdHash()
Private txt As INPUTElementImpl = CType(IIf(.TypeOf htmlelements("txt") Is
INPUTElementImpl, htmlelements("txt"), Nothing), INPUTElementImpl)
Private specialEvents As String() = New String(1) {}
Private specialEvents(0) = "Yes"
Private specialEvents(1) = "No"
MessageBox.Show("Before Merging:" + Me.txt.SupportedEvents.Length.ToString())
Me.txt.MergeSupportedEvents(specialEvents)
MessageBox.Show("After Merging:" + Me.txt.SupportedEvents.Length.ToString())

```

4.5.1 Element Types

The following are the various HTML elements supported by Essential HTMLUI.

Anchor Element	Body Element	Bold Element	BR Element	Code Element	Custom Element	Div Element
EM Element	Font Element	Form Element	H1 - H6 Element	Head Element	HR Element	HTML Element
I Element	IMG Element	Input Element	LI Element	Link Element	OL Element	P Element
PRE Element	Script Element	Select Element	Span Element	Strong Element	Style Element	Table Element
TD Element	TextArea Element	TH Element	TR Element	U Element	UL Element	

4.5.2 A - Anchor Element

The **A** element is used in creating links to another document or in creating bookmarks within the same document. This element is defined by the **<a>** tag in the HTML code. The **AELEMENTImpl** class contains the properties and methods related to this element. Some of the important properties and methods are listed below:

Properties

- **IsVisited**: Gets a bool value (either true / false) indicating whether the link is visited or not. This may be used in changing the color of the links visited.
- **HoverFormat**: Gets the format of the **A** element when the user hovers the mouse pointer over the link.
- **VisitedFormat**: Gets the format of the **A** element visited recently.

[C#]

```
// IsVisited property gets the Boolean value indicating whether the link is
visited or not
// VisitedFormat property get the format of the A element visited recently.
Hashtable htmlElements =
this.htmluiControl1.Document.GetElementsByIdHash();
AElementImpl a = htmlElements["a"] as AElementImpl;
this.label1.Text = "\nA(IsVisited and VisitedFormat):" +
this.a.IsVisited.ToString() + ", " +
this.a.VisitedFormat.BackgroundColor.Name.ToString();
```

[VB.NET]

```
' IsVisited property gets the Boolean value indicating whether the link is
visited or not
' VisitedFormat property get the format of the A element visited recently.
Private htmlElements As Hashtable =
Me.HtmluiControl1.Document.GetElementsByIdHash()

Private a As AElementImpl = CType(If(TypeOf htmlElements("a") Is AElementImpl,
htmlElements("a"), Nothing), AElementImpl)

Private Me.label1.Text=Constants.vbLf & "A(IsVisited and VisitedFormat):" &
Me.a.IsVisited.ToString()+" "+Me.a.VisitedFormat.BackgroundColor.Name.ToString(
)
```

Methods

- **ResetVisited**: Excludes the element from the list containing the visited links.

4.5.3 B - Bold Element

The **B** element is responsible for formatting the specified text in bold style. The **BElementImpl** class contains the properties and methods of this element. The **SUBElementImpl** and **SUPElementImpl** classes are also responsible for bolding elements. They also contain the properties and methods for the element's behavior.

4.5.4 BODY Element

The **BODY** element forms the main section in the HTMLUI because this element contains all the other elements and details regarding their position and properties. The **BODYElementImpl** class contains the properties and methods for this element.

4.5.5 BR - Break Element

The **BR** element is used for inserting a line break after a particular line. This is implemented using the **
** tag in the HTML document. The **BRElementImpl** class contains the properties and methods for this element's behavior.

Properties

- **IsVisible:** Gets / sets a boolean value to indicate whether the control is shown / hidden.

[C#]

```
// Get the Boolean value to indicate whether the control is visible or not.
Hashtable htmlelements =
this.htmluiControl1.Document.GetElementsByUserIdHash();
BRElementImpl br = htmlelements["br"] as BRElementImpl;
this.labell1.Text = "\nBR(IsVisible):" + this.br.IsVisible.ToString();
```

[VB.NET]

```
' Get the Boolean value to indicate whether the control is visible or not.
Private htmlelements As Hashtable =
Me.HtmluiControl1.Document.GetElementsByUserIdHash()
Private br As BRElementImpl = CType(If(OfType htmlelements("br") Is
BRElementImpl, htmlelements("br"), Nothing), BRElementImpl)
Me.labell1.Text = Constants.vbLf & "BR(IsVisible):" & Me.br.IsVisible.ToString()
```

4.5.6 CODE Element

The **CODE** element is used in marking the specified text as a computer code, formatted using monospaced font. It uses the **CODEElementImpl** class which contains the properties and methods determining the element's behavior.

4.5.7 CUSTOM Element

The **CUSTOM** element is used in creating the custom tags as determined by the user. The **CUSTOMElementImpl** class is responsible for the custom controls declared by the **<custom>** tag in the HTML document and contains the element's properties and methods.

4.5.8 DIV - Division Element

The **DIV** element divides the given page into logical sections. The **DIVElementImpl** class contains the properties and methods that describe the behavior of the DIV element.

4.5.9 EM - Emphasize element

The **EM** element is responsible for emphasizing a specific text, usually in italics. The HTML code uses the **** tag to specify this element. The **EMElementImpl** class specifies the properties and methods to be used while coding this element.

4.5.10 FONT Element

The **FONT** element is used for changing the font face, size and color of the specified text. The **** tag is used to specify the FONT element in the HTML code. The **FontElementImpl** class determines the properties and methods used by this element.

4.5.11 FORM Element

The **FORM** element creates a form for user input. A form can contain text fields, check boxes, radio buttons and other form fields. The **FORMElementImpl** class is used to determine the properties and methods of this element.

4.5.12 H1 - H6 Header Elements

- The **H1** element is used to define a header. The H1 element produces the largest header. The **H1ElementImpl** class is used in determining the properties and methods of the H1 element.
- The **H2** element is used to define a header. The **H2ElementImpl** class is used in determining the properties and methods of the H2 element.
- The **H3** element is used to define a header. The **H3ElementImpl** class is used in determining the properties and methods of the H3 element.
- The **H4** element is used to define a header. The **H4ElementImpl** class is used in determining the properties and methods of the H4 element.
- The **H5** element is used to define a header. The **H5ElementImpl** class is used in determining the properties and methods of the H5 element.

- The **H6** element is used to define a header. The **H6** element produces the smallest header. The **H6ElementImpl** class is used in determining the properties and methods of the H6 element.

4.5.13 HEAD Element

The **HEAD** element contains information regarding the document like the title of the document, links to the style sheets, and so on. The **HEADElementImpl** class is used to specify the properties and methods of the head element.

4.5.14 HTML Element

The **HTML** element is used to specify that the document is an HTML document. It is mentioned by the **<html>** tag in the HTML document. The properties and methods of this element are defined in the **HTMLElementImpl** class.

4.5.15 HR - Horizontal Rule Element

The **HR** element is used in creating horizontal rules. The **HRElementImpl** class contains the methods and properties for this element.

4.5.16 I - Italics Element

The **I** element is used in formatting the specified text in italics. The **IElementImpl** class is used in determining the properties and methods for the I element.

4.5.17 IMG - Image Element

The **IMG** element is used in defining and applying an image in the document wherever needed. The **** tag is used for this. The properties and methods of this element is defined in the **IMGElementImpl** class.

Properties

- **Image**: Gets the bitmap of the image that represents this element

[C#]

```
// Gets the bitmap of the image that represents IMG element.
Hashtable htmlelements =
this.htmluiControl1.Document.GetElementsByIdHash();
IMGElementImpl img = htmlelements["img"] as IMGElementImpl;

// Gets the width and height of the image.
this.labell1.Text = "\nIMG(Image)" + this.img.Image.PhysicalDimension.ToString();
```

[VB.NET]

```
' Gets the bitmap of the image that represents IMG element.
Private htmlelements As Hashtable =
Me.htmluiControl1.Document.GetElementsByIdHash()
Private img As IMGElementImpl = CType(If(OfType htmlelements("img") Is
IMGElementImpl, htmlelements("img"), Nothing), IMGElementImpl)
Me.labell1.Text = Constants.vbLf & "IMG(Image)" &
Me.img.Image.PhysicalDimension.ToString()
```

4.5.18 INPUT Element

The **INPUT** element is used for getting input from the user. It can be a text box, a button element or a check box which is determined by the type attribute of the **<input>** tag in the HTML document. The **INPUTElementImpl** class is used in determining the methods and properties for this element.

Properties

- **UserControl:** Gets / sets the user control instance for the particular input element declared by the user

[C#]

```
// Sets the user control instance for the particular input element declared by
the user.
Hashtable htmlelements =
this.htmluiControl1.Document.GetElementsByIdHash();
INPUTElementImpl txt = htmlelements["txt"] as INPUTElementImpl;
this.txt.UserControl.CustomControl.Text = "This is a textBox";
```

[VB.NET]

```
' User control property sets the user control instance for the particular
input element declared by the user.
Private htmlelements As Hashtable =
Me.HtmluiControl1.Document.GetElementsByIdHash()
```



```
Private txt As INPUTElementImpl = CType(IIf(.TypeOf htmlelements("txt") Is
INPUTElementImpl, htmlelements("txt"), Nothing), INPUTElementImpl)
Me.txt.UserControl.CustomControl.Text = "This is a textBox"
```

Methods

- **InfillFromXMLElement:** Detects the type of control from the **type** attribute and creates that control

4.5.19 LI - List Element

The **LIST** element is used to define a list item. The **LIElementImpl** class is used to determine the properties and methods for this element. There are two types of lists that are supported by HTMLUI.

- [OL Element](#): Ordered List Element
- [UL Element](#): Unordered List Element

4.5.20 LINK Element

The **LINK** element is used to define links to other documents, style sheets, and so on. The **LinkElementImpl** is used to determine the methods and properties for the link element.

Properties

- **IsVisible:** Gets / sets a value indicating whether the link is shown / hidden

[C#]

```
// Get the value indicating whether the link is visible or not.
Hashtable htmlelements =
this.htmluiControl1.Document.GetElementsByIdHash();
LinkElementImpl link = htmlelements["link"] as LinkElementImpl;
this.labell1.Text = "\nLink(IsVisible):" + this.link.IsVisible.ToString();
```

[VB.NET]

```
' Get the value indicating whether the link is visible or not.
Private As Hashtable = Me.htmluiControl1.Document.GetElementsByIdHash()
Private link As LinkElementImpl = CType(IIf(.TypeOf HtmlElement("link") Is
LinkElementImpl, htmlelements("link"), Nothing), LinkElementImpl)
Me.labell1.Text = Constants.vbLf & "Link(IsVisible):" &
Me.link.IsVisible.ToString()
```

Methods

- **GetCssStream**: Returns a stream CSS data of the link element

4.5.21 OL - Ordered List Element

The **OL** element is used in generating an ordered list as specified by the user. The properties and methods of this element are defined in the **OLElementImpl** class.

4.5.22 P - Paragraph Element

The **P** element is used to define a paragraph in the document. The user can determine the properties and methods for the **P** element by invoking the **PElementImpl** class.

4.5.23 PRE - Preformatted Element

The **PRE** element defines preformatted text. The text enclosed in the pre element usually preserves the spaces and line breaks. The enclosed text appears exactly as in the HTML document. The properties and methods for this element can be determined from the **PREElementImpl** class.

4.5.24 SCRIPT Element

The **SCRIPT** element is used to define scripts to the HTML document. This makes the document self-contained. It does not require any other external ways to define the operation of the document's elements. The **SCRIPTElementImpl** class is used to determine the properties and methods for this element.

Properties

- **IsVisible**: Gets / sets a value indicating whether the script is shown / hidden

```
[C#]

// Gets or sets a value indicating whether the script is visible or not.
Hashtable htmlelements =
this.htmluiControll.Document.GetElementsByUserIdHash();
this.script = htmlelements["script"] as SCRIPTElementImpl;
this.labell1.Text = "\nScript(IsVisible):" + this.script.IsVisible.ToString();
```

[VB.NET]

```
' Gets or sets a value indicating whether the script is visible or not.
Private htmlelements As Hashtable =
Me.HtmluiControl1.Document.GetElementsByIdHash()
Private Me.script = CType(IIf(OfType htmlelements("script") Is
SCRIPTElementImpl, htmlelements("script"), Nothing), SCRIPTElementImpl)
Private Me.label1.Text = Constants.vbLf & "Script(IsVisible):" &
Me.script.IsVisible.ToString()
```

Methods

- **GetScriptCode:** Gets the string format of the script code

[C#]

```
MessageBox.Show("ScriptCode:\n" + this.script.GetScriptCode().ToString());
```

[VB.NET]

```
MessageBox.Show("ScriptCode:" &
Constants.vbLf+Me.script.GetScriptCode().ToString())
```

4.5.25 SELECT Element

The **SELECT** element is used to define a drop-down list. The user can specify the number of items to include in the drop-down list. The **SELECTElementImpl** class defines the properties and methods for this element.

Properties

- **UserControl:** Gets / sets the user control instance to the particular element

[C#]

```
// UserControl property gets or sets the user control instance to the
particular element.
Hashtable htmlelements =
this.htmluiControl1.Document.GetElementsByIdHash();
this.select = htmlelements["select"] as SELECTElementImpl;
this.label1.Text = "\nSelect(UserControl):" +
this.UserControl.DefaultSize.ToString();
```

[VB.NET]

```
' UserControl property gets or sets the user control instance to the
particular element.
```

```

Private htmlElements As Hashtable =
Me.HtmluiControl1.Document.GetElementsByUserIdHash()
Me.Select = CType(If(TypeOf htmlElements("select") Is SELECTElementImpl,
htmlElements("select"), Nothing), SELECTElementImpl)
Me.label1.Text = Constants.vbLf & "Select(UserControl):" &
Me.Select.UserControl.DefaultSize.ToString()

```

Methods

- **InfillFromXMLElement:** Detects the type of control and creates the particular control

4.5.26 SPAN Element

The **SPAN** element is used to group inline elements in the document and create custom character styles. The **SPANElementImpl** class is used to determine the properties and methods for the span element.

4.5.27 STRONG Element

The **STRONG** element is used to emphasize the specified text, usually in bold. The **STRONGElementImpl** class is used to define the properties and methods for the strong element.

4.5.28 STYLE Element

The **STYLE** element is used to implement custom style in a document. It occurs inside the head section. An external style sheet is linked by using the **<link>** tag in a HTML document. The **StyleElementImpl** class is invoked for defining the properties and methods of the style element.

Properties

- **IsVisible:** Gets / sets a value indicating whether the link is shown / hidden

[C#]

```

// Gets a value indicating whether the link is visible or not.
Hashtable htmlElements =
this.htmluiControl1.Document.GetElementsByUserIdHash();
StyleElementImpl link = htmlElements["style"] as StyleElementImpl;
this.label1.Text = "\nLink(IsVisible):" + link.IsVisible.ToString();

```

[VB.NET]

```

' Gets a value indicating whether the link is visible or not.

```

```

Private htmlelements As Hashtable =
Me.HtmluiControl1.Document.GetElementsByUserIdHash()
Private link As StyleElementImpl = CType(IIf(.TypeOf htmlelements("style") Is
StyleElementImpl, htmlelements("style"), Nothing), StyleElementImpl)
Private Me.label1.Text = Constants.vbLf & "Link(IsVisible):" &
link.IsVisible.ToString()

```

Methods

- **GetCssStream:** Returns a stream of inner CSS data of the style element

4.5.29 TABLE Element

The **TABLE** element is used to create tables in a document. The table element contains the **TR**, **TD** elements within it. The **TABLEElementImpl** class is used to determine the properties and methods for the table element.

Properties

- **ColsCount:** Gets / sets the number of columns present in the table
- **RowCount:** Gets / sets the number of rows present in the table

[C#]

```

// Gets the number of columns and rows present in the table.
Hashtable htmlelements =
this.htmluiControl1.Document.GetElementsByUserIdHash();
TABLEElementImpl table = htmlelements["table"] as TABLEElementImpl;
this.label1.Text = "\nTable(ColsCount and RowCount):" +
table.ColsCount.ToString() + "," + table.RowCount.ToString();

```

[VB.NET]

```

' Gets the number of columns and rows present in the table.
Private htmlelements As Hashtable =
Me.HtmluiControl1.Document.GetElementsByUserIdHash()
Private table As TABLEElementImpl = CType(IIf(.TypeOf htmlelements("table") Is
TABLEElementImpl, htmlelements("table"), Nothing), TABLEElementImpl)
Private Me.label1.Text = Constants.vbLf & "Table(ColsCount and RowCount):" &
table.ColsCount.ToString()+","+table.RowCount.ToString()

```

4.5.30 TD - Table cell Element

The **TD** element is used to create regular cells inside a table. The **TDElementImpl** class contains the properties and methods for the table element.

4.5.31 TEXTAREA Element

The **TEXTAREA** element is used to define a multiline textbox, allowing the user to enter unlimited characters. The **TEXTAREAElementImpl** class is invoked to define the properties and methods of the element.

Properties

- **UserControl:** Gets / sets the user control instance for the particular input element declared by the user

[C#]

```
// UserControl property gets the user control instance for the particular input
// element declared by the // user.
Hashtable htmlElements =
this.htmluiControl1.Document.GetElementsByIdHash();
TEXTAREAElementImpl txt = htmlElements["txt"] as TEXTAREAElementImpl;
this.txt.UserControl.CustomControl.Text = "This is a multiline textBox";
```

[VB.NET]

```
' UserControl property gets the user control instance for the particular input
' element declared by the ' user.
Private htmlElements As Hashtable =
Me.HtmluiControl1.Document.GetElementsByIdHash()
Private txt As TEXTAREAElementImpl = CType(If(OfType(htmlElements("txt")) Is
TEXTAREAElementImpl, htmlElements("txt"), Nothing), TEXTAREAElementImpl)
Private Me.txt.UserControl.CustomControl.Text= "This is a multiline textBox"
```

4.5.32 TH - Table Head Element

The **TH** element is used to create header cells inside a table. The **THElementImpl** class contains the properties and methods for the table header element. The text inside this element is formatted in bold, by default.

4.5.33 TR - Table Row Element

The **TR** element is used to create rows inside a table. The **TRElementImpl** class contains the properties and methods for the table cell coding.

Properties

- **CellsCount**: Gets the number of cells present in the row
- **VirtualCellsCount**: Gets the total number of cells including the Colspan

[C#]

```
// Gets the number of cells present in the row and gets total number of cells
including the colspan.
Hashtable htmlelements =
this.htmluiControll.Document.GetElementsByIdHash();
TRElementImpl tr = htmlelements["tr"] as TRElementImpl;
this.labell1.Text = "TR(CellsCount and VirtualCellsCount):"+
tr.CellsCount.ToString()+","+" tr.VirtualCellsCount.ToString();
```

[VB.NET]

```
' Gets the number of cells present in the row and gets total number of cells
including the colspan.
Private htmlelements As Hashtable =
Me.htmluiControll.Document.GetElementsByIdHash()
Private tr As TRElementImpl = CType(IIf(OfType htmlelements("tr") Is
TRElementImpl, htmlelements("tr"), Nothing), TRElementImpl)
Private Me.labell1.Text = "TR(CellsCount and VirtualCellsCount):"+
tr.CellsCount.ToString()+","+" tr.VirtualCellsCount.ToString()
```

4.5.34 UL - Unordered List Element

The **UL** element is used in generating an unordered list. The properties and methods of this element are defined in the **ULElementImpl** class.

4.5.35 U - Underline Element

The **U** element is used to underline the specified text. The **UElementImpl** class contains the properties and methods for the underline element.

4.6 HTML Tags

This sections details the HTML tags supported by HTMLUI. Most of the tags conform to the **XHTML** standard. Some of the tags support additional functionality implemented through custom attributes. Since HTMLUI considers each HTML tag as an XML element, it is recommended to use closing tags for each HTML element at the end. These tags and attributes are also marked and explained in this section.

4.6.1 A - Anchor Tag

The **Anchor** tag is used for creating links to other files or in creating bookmarks. This tag ends with ``. It includes the following attributes.

- **href**: Specifies the URL of the page to which the link is to be made
- **rel**: Specifies the relation between the current document and the target url

The following example illustrates how the Anchor tag is rendered in HTMLUI.

[C#]

```
string htmlCode="<html><head><title>A Tag support</title></head><body><A href='link.htm'>Link</A></body></html>";
this.htmluiControl1.LoadFromString(htmlCode);
```

[VB.NET]

```
Private htmlCode As String = "<html><head><title>A Tag support</title></head><body><A href='link.htm'>Link</A></body></html>"
Me.HtmluiControl1.LoadFromString(htmlCode)
```

4.6.2 ABBR - Abbreviation Tag

The **Abbreviation** tag is used to indicate the abbreviated forms of long and important texts. The **title** attribute is used to display a tooltip text when the cursor is moved over the abbreviated text. The support for `<abbr>` tag is shown in the following code snippet.

[HTML]

```
File Location and Name: C:\MyProjects\Anchor\abbr.html

<html>
  <body>
    <abbr title="United Nations">UN</abbr>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\Anchor\abbr.html");
```

[VB.NET]


```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\Anchor\abbr.html")
```

4.6.3 Acronym Tag

The **Acronym** tag is used to define the start of an acronym. The HTMLUI control supports the acronym tag which helps in providing useful tips to the users when marked up. The **title** attribute is used to provide a tooltip text when the mouse pointer moves over the acronym.

[HTML]

File Location and Name: C:\MyProjects\Acronym\acronym.html

```
<html>
  <body>
    <abbr title="World Wide Web">WWW</abbr>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\Acronym\acronym.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\Acronym\acronym.html")
```

4.6.4 Comment Tag

HTMLUI control supports the use of HTML Comment tags while developing applications. The **Comment** tag is used to include a brief description by the developer, which helps the user to understand the code and also helps to edit the code at a later date. Normally the content inside the comment is ignored by the HTMLUI control.

[HTML]

File Location and Name: C:\MyProjects\Comment\comment.html

```
<html>
<body>
  <!-- This is a HTML comment -->
  <p>HTMLUI supports HTML commenting</p>
</body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\Comment\comment.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\Comment\comment.html")
```

4.6.5 Font Style Tags

The **Font Style** tags are used to format the appearance of the specified text when rendered in HTMLUI. The following are the font style tags supported in HTMLUI:

- ****: Bold tag renders the text in bold face
- **<i>**: Italics tag renders italicized text
- **<u>**: Underline tag renders underlined text
- ****: Emphasizing Text tag highlights important text in the document
- ****: Strong tag renders specified text in bold face
- **<code>**: Code tag renders specified text similar to computer coded text

[HTML]

File Location and Name: C:\MyProjects\FontStyle\fontStyle.html

```
<html>
  <body>
    <b>Bold text</b><br/>
    <i>Italic text</i><br/>
    <u>Underlined text</u><br/>
    <em>Emphasized Text</em><br/>
    <strong>Strong Text</strong><br/>
    <code>Computer coded text</code>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\FontStyle\fontStyle.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\FontStyle\fontStyle.html")
```

4.6.6 BODY - Body Tag

The **Body** element defines the body of the HTML document. This is the parent element of all the HTML elements visible in the browser. The contents to be displayed in the HTMLUI control are placed inside the body tag. In HTMLUI the body element is considered as the basis for all the elements present in the document. The following code snippet shows the use of a body element in rendering HTML documents in HTMLUI control.

[HTML]

File Location and Name: C:\MyProjects\body\bodyTag.html

```
<html>
  <body bgcolor="#ffffff">
    <p>Body tag holds the contents to be displayed in the browser.</p>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\body\bodyTag.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\body\bodyTag.html")
```

4.6.7 BR - Break Tag

The **Break** tag is used to insert a line break in the document displayed.

[HTML]

File Location and Name: C:\MyProjects\break\br.html

```
<html>
  <body>
    <p>This is the first line.</p><br/><p>This is the next line</p>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\break\br.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\break\br.html")
```

4.6.8 DIV - Division Tag

The **Division** tag is used to define a division or a section defined in the document that is rendered in the HTMLUI control. The **align** attribute of the div tag is used to align the text displayed inside the div tag. The align attribute uses the following values to specify the position of the text display.

- left
- Right
- center
- justify

[HTML]

File Location and Name: C:\MyProjects\divide\div.html

```
<html>
  <body>
    <div align="left">Essential Studio</div>
    <div align="right">Tools</div>
    <div align="center">Grid</div>
    <div align="justify">HTMLUI</div>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\divide\div.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\divide\div.html")
```

4.6.9 FORM - Form Tag

The **Form** tag is used to create a form-based dialog for the users to give their inputs. The forms input element may be a text box, button, radio button, check box, and so on, based on the necessity of the user.

[HTML]

File Location and Name: C:\MyProjects\UserInput\form.html

```
<html>
  <body>
    <form>
      <input type="text"/><br/>
      <input type="submit"/><br/>
      <input type="radio"/><br/>
      <input type="checkbox"/><br/>
    </form>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\UserInput\form.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\UserInput\form.html")
```

4.6.10 HEAD - Head Tag

The **Head** element contains the information required for processing the document. The contents of the head tag will not be displayed in the HTMLUI control. The HTMLUI control receives information only through the head element. The following are the information obtained:

- Link reference to the style sheets by using the Link tag of the head section
- Styles to the html elements by using the Style tag of the head section
- Title for the document in the Title tag of the head section

[HTML]

File Location and Name: C:\MyProjects\heading\head.html

```
<html>
  <head>
    <link href="style.css" rel="stylesheet" type="text/css">
    <style>p{"background-color:#dae5f5;"}</style>
    <title>Head section</title>
  </head>
  <body>
    <p>The head contains the title for the document</p>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\heading\head.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\heading\head.html")
```

4.6.11 H1 - H6 Header Tag

The **Header** tags are defined as the h tags. These header tags are classified into six types based on the font size. HTMLUI control supports all the header tags ranging from h1 - the largest header to h6 - the smallest header.

[HTML]

File Location and Name: C:\MyProjects\header\h.html

```
<html>
  <body>
    <h1 align="center">Header 1</h1><br/>
    <h2 align="left">Header 2</h2><br/>
    <h3 align="justify">Header 3</h3><br/>
    <h4 align="right">Header 4</h4><br/>
    <h5>Header 5</h5><br/>
    <h6>Header 6</h6><br/>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\header\h.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\header\h.html")
```

4.6.12 HR - Horizontal Rule Tag

The **Horizontal Rule** tag is used to draw an horizontal line in the document. The <hr> tag in HTMLUI supports the following attributes.

- **width:** Specifies the width of the line

- **size**: Specifies the thickness or height of the line
- **noshade**: Renders the specified line in a solid color

[HTML]

File Location and Name: C:\MyProjects\HorizRule\rule.html

```
<html>
  <body>
    <p>Syncfusion</p><hr width=100 size=10 noshade /><p>Essential Studio</p>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\HorizRule\rule.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\HorizRule\rule.html")
```

4.6.13 HTML - HTML Tag

The **HTML** tag specifies that the document to be loaded in the HTMLUI control is an HTML document. This tag contains the head and the body elements as its child elements.

[HTML]

File Location and Name: C:\MyProjects\HTML\htmlElement.html

```
<html>
  <head>
    <title>New html document</title>
  </head>
  <body>
    <p>The HTML tag contains the head and the body elements as its child
    element.</p>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\HTML\htmlElement.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\HTML\htmlElement.html")
```

4.6.14 IMG - Image Tag

The **Image** tag is used to display an image in the HTMLUI control. The HTMLUI control supports the following attributes for the image element, which can be used in developing advanced HTMLUI applications.

- **src:** Source attribute is used to access the location of the image file. The image may be located in the local folder or in the URI.
- **alt:** Alternate Text tag provides a tooltip text when the mouse is moved over the image. In case the image is not rendered in the control, this text is displayed in place of the image.

[HTML]

File Location and Name: C:\MyProjects\img\imageElement.html

```
<html>
  <body>
    
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\img\imageElement.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\img\imageElement.html")
```

4.6.15 INPUT - Input Tag

The **Input** tag is used to receive some inputs from the user. The input tag uses the following attributes to provide a value for the specified input elements from the user.

- **type:** Specifies the type of input control to be placed in the document. HTMLUI control supports the following input elements:

Attribute Value	Control that will be Displayed
Text	Text Box

Button	Button
CheckBox	Check Box
Radio	Radio Button
Password	Password Box
Reset	Reset Button
Submit	Submit Button

- **value:** Specifies the default text that will appear on the control after being rendered on the document
- **size:** Specifies the size of the input document
- **name:** Specifies a unique name to the control. In HTMLUI the **Control.Name** property will access the name given to the control in code and not the value of this name attribute. The user has to access this value with the help of the **Control.Attributes["name"].Value** property. This will return the value of this attribute.
- **maxlength:** Specifies the maximum number of characters that can be displayed inside the text fields
- **disabled:** Disables the control. Any change that the user makes in the control will not be updated in the control.
- **checked:** Displays the checkbox or the radio button selected by default in the document

[HTML]

File Location and Name: C:\MyProjects\input\input.html

```
<html>
  <body>
    <input type="text" value="textbox" size="20" maxlength="5" /><br/>
    <input type="button" value="button element"/><br/>
    <input type="checkbox" value="checkbox" checked/><br/>
    <input type="password" value="password" size="20"/><br/>
    <input type="radio" value="radio"/><br/>
    <input type="reset" value="reset"/><br/>
    <input type="submit" value="submit" size="50"/><br/>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\input\input.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\input\input.html")
```

4.6.16 LI - List Item Tag

The **List Item** tag defines a list item. The `` tag is used inside the `` tag or `` tag to define each and every list item included inside them.

[HTML]

File Location and Name: C:\MyProjects\listItem\li.html

```
<html>
  <body>
    <ol>
      <li>Essential Tools</li>
      <li>Essential Chart</li>
      <li>Essential HTMLUI</li>
    </ol>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\listItem\li.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\listItem\li.html")
```

4.6.17 LINK - Link Tag

The **Link** tag is used to link another document to the current HTML document. This tag is generally used to link an external CSS style sheet to the HTML document. The following attributes for the link tag are supported in the HTMLUI control:

- **rel**: Specifies the relationship between the two documents
- **type**: Specifies the type of the document to be linked, either text or image
- **href**: Specifies the location of the document to be linked to the current document

[HTML]

File Location and Name: C:\MyProjects\link\link.html

```
<html>
  <head><link rel="stylesheet" type="text/css" href="background.css"/></head>
  <body>
    <p>The document receives its background color from the external style
    sheet.</p>
  </body>
</html>
```

[Stylesheet]

File Location and Name: C:\MyProjects\link\background.css

```
body
{
  background-color: #dae5f5;
}
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\link\link.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\link\link.html")
```

4.6.18 OL - Ordered List Tag

The **Ordered List** tag defines the start of an ordered list. The ordered list is numbered in the following scheme as '1, i, I, a, A' based on the requirements of the user. The ordered list contains the items as its child elements. Each element defines a list item. The type attribute is used to choose the required numbering style.

[HTML]

File Location and Name: C:\MyProjects\listItem\ol.html

```
<html>
<body>
  <ol type="i">
    <li>Essential Tools</li>
    <li>Essential Chart</li>
    <li>Essential HTMLUI</li>
  </ol>
</body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\listItem\ol.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\listItem\ol.html")
```

4.6.19 OPTION - Option Tag

The **Option** tag is used to include items to the drop-down list. The option tag is the child of the select tag. The option tag in HTMLUI supports the **selected** attribute. This attribute specifies the control that the specified option item is to be selected at startup.

[HTML]

File Location and Name: C:\MyProjects\select\option.html

```
<html>
  <body>
    <p>Dear Customer, please choose your option among our following
    products.</p>
    <select>
      <option>Essential Tools</option>
      <option>Essential Chart</option>
      <option selected="selected">Essential HTMLUI</option>

      <!-- HTMLUI also supports this format, <option selected>Essential
      HTMLUI</option> -->
    </select>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\select\option.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\select\option.html")
```

4.6.20 P - Paragraph Tag

The **P** tag is used to define a paragraph in the HTML document. HTMLUI supports the P tag along with its align attribute. The **align** attribute is used to specify the alignment of the text within the paragraph. The alignment values are as follows:

- **left**: Left-aligns the text
- **right**: Right-aligns the text
- **center**: Positions text at the center of the document
- **justify**: Left-aligns and right-aligns text

[HTML]

File Location and Name: C:\MyProjects\paragraph\p.html

```
<html>
  <body>
    <p align="left"><b>Essential Studio</b></p>
    <p align="justify">Essential Studio contains 10 .NET libraries. It has
      components ranging from a Grid control to a HTML display control. They are
      available for both Windows Forms and ASP.NET.</p>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\paragraph\p.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\paragraph\p.html")
```

4.6.21 PRE - Preformatted Tag

The HTMLUI control renders preformatted texts written inside the <pre> tags. These texts are rendered with the fixed-pitch fonts and the control preserves the spaces and the line breaks used in the document.

[HTML]

File Location and Name: C:\MyProjects\pre\pre.html

```
<html>
  <body>
    <pre>
      Evaluation Center:
      The Evaluation Center has online sample code and complete sample downloads
      to help you get started quickly.
```

```
</pre>
</body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\paragraph\p.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\paragraph\p.html")
```

4.6.22 SCRIPT - Script Tag

The **Script** tag is used to include executable user defined code snippets inside the HTML document. This is to make the HTML document self-contained and not to depend on external means for executing an action.

HTMLUI supports C#, Visual Basic and Jscript. The **language** attribute is used to choose the script language from the supported types.

[HTML]

File Location and Name: C:\MyProjects\scripting\scripting.html

```
<html>
<head><title>Scripting</title></head>
<body>
<P><input type="button" id="btn"></p>
<p><img id="img" src=""/></p>
<script language="C#">
```

```
using System;
using System.Windows.Forms;
using System.Collections;
using Syncfusion.Windows.Forms.HTMLUI;
```

```
namespace Syncfusion
{
public class Script
    {
        IHTMLElement button = null;
        IHTMLElement img = null;
        Hashtable htmelements = new Hashtable();
        /* Initializes script.*/
        public static Script OnScriptStart()
        {
```

```

        return new Script();
    }
    public Script()
    {
        if( Global.Document == null )
        {
            MessageBox.Show( "Document is NULL" );
            return;
        }
        button = Global.Document.GetElementById( "btn" );
        img = Global.Document.GetElementById( "img" );
        button.Click += new EventHandler(button_Click);
    }
    public void button_Click(object sender, EventArgs ar)
    {
        img.Attributes["src"].Value="sync.jpg";
    }
}
</script><br/>
</body>
</html>

```

4.6.23 SELECT - Select Tag

The **Select** tag is used to create a drop-down list of options that can be selected by the user to give some input to the application. The select tag in HTMLUI supports the following attributes that improve the usage of the application.

- **size:** Specifies the number of items to be displayed in the select control in the document
- **disabled:** Displays a disabled list box in which no items can be selected

[HTML]

File Location and Name: C:\MyProjects\select\select.html

```

<html>
  <body>
    <p>
      Essential Studio includes ten component libraries in one great package.
      Each of these products has a unique and useful feature set.
    </p>
    <select size="2">
      <option>Essential Tools</option>
      <option>Essential Chart</option>
      <option>Essential Grid</option>
      <option>Essential HTMLUI</option>
    </select>
  </body>
</html>

```

```
</select>
</body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\select\select.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\select\select.html")
```

4.6.24 SPAN - Span Tag

The **Span** element is used to group inline elements present inside the element tags in the document.

[HTML]

File Location and Name: C:\MyProjects\span\span.html

```
<html>
<body>
  <p>100% .NET HTML display engine that can be used to create extremely
  flexible user interfaces. Part of <span style="color:#0000FF;">Essential
  Studio</span> Enterprise.</p>
</body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\span\span.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\span\span.html")
```

4.6.25 STYLE - Style Tag

The **Style** tag is used to apply styles to the HTML elements in the HTML document. The style tag is placed inside the head section and it will not be visible in the document. Styles can be applied to the HTML elements in the following ways:

1. **By specifying the Tag Names:** Specifies styles by writing CSS styles with the tag name inside the Style tag

[HTML]

```
<style>p {color: red}</style>
<p>Sample</p>
```

2. **By specifying the Class Names to the Styles:** Specifies styles with the help of the class names inside the Style tag

[HTML]

```
<style>.span {color: green}</style>
<span class="span">Sample</span>
```

3. **By specifying ID Class Selectors:** Specifies styles by writing styles with the unique id inside the Style tag, and assigning them to the HTML elements by using the **id** attribute

[HTML]

```
<style>#divide {color: blue}</style>
<div id="divide">Sample</span>
```

Style tag in HTMLUI also supports the **type** attribute. The type attribute is optional and it specifies the type of the content in the HTML document.

[HTML]

File Location and Name: C:\MyProjects\style\style.html

```
<html>
<head>
  <style type="text/css">
    p {color: red}
    .span {color: blue}
    #bluebg { background-color: #dae5f5; background-repeat: repeat-x;}
  </style>
</head>
<body id="bluebg">
  <p>100% .NET HTML display engine that can be used to create
  extremely flexible user interfaces. Part of <span
  class="span">Essential Studio</span> Enterprise.</p>
</body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\style\style.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\style\style.html")
```

4.6.26 SUB - Subscript Tag

The **Subscript** tag defines a subscript text. The use of subscript text in HTMLUI is shown below.

[HTML]

File Location and Name: C:\MyProjects\sub\sub.html

```
<html>
  <head>
  </head>
  <body>
    Water - H<sub>2</sub>O<br/>
    Sulphuric Acid - H<sub>2</sub>SO<sub>4</sub><br/>
    Carbon-di-Oxide - CO<sub>2</sub>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\sub\sub.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\sub\sub.html")
```

4.6.27 SUP - Superscript Tag

The **Superscript** tag defines a superscript text. The use of superscript text in HTMLUI is shown below.

[HTML]

File Location and Name: C:\MyProjects\sup\sup.html

```
<html>
  <head>
</head>
  <body>
    Pythagoras theorem:<br/>
    In a right angled triangle,<br/>
    <code>
      hypotenuse<sup>2</sup> = opposite<sup>2</sup> + adjacent<sup>2</sup>
    </code>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\sup\sup.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\sup\sup.html")
```

4.6.28 TABLE - Table Tag

A **Table** tag defines a table in an HTML document. The table tag uses the <tr> tag to define a row and a <td> tag to define a cell element. The HTMLUI control supports the table with the following attributes that helps the rendering and display of complex html pages in the control easily.

- **bgcolor:** Specifies the background color of the table
- **border:** Specifies the thickness of the table border

[HTML]

File Location and Name: C:\MyProjects\table\table.html

```
<html>
  <body>
    <table bgcolor="#dae5f5" border="1">
      <tr>
        <td>Sample</td><td>Sample</td>
      </tr>
    </table>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\table\table.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\table\table.html")
```

4.6.29 TD - Table Cell Tag

The **Table Cell** tag defines a cell inside a table. The **<td>** tag has a parent **<tr>** tag to define the row and a **<table>** tag to define the table in which it is present. The **td** tag in HTMLUI supports the following attributes that help the user in designing custom structures for their documents easily.

- **align**: Specifies the alignment of the text inside the table cell
- **bgcolor**: Specifies a background color for the specified cell
- **colspan**: Spans the cell to the specified number of columns. This is used in merging the columns in the table.
- **height**: Specifies custom height for the cells
- **nowrap**: Extends the text inside a particular cell into a single line. This display extends the width of the cell according to the contents inside it.
- **rowspan**: Extends the height of the cell to the specified number of rows. This is helpful in custom merging the rows of the given cell
- **valign**: Determines the vertical alignment of the text inside the table cell
- **width**: Specifies user-defined width for the specified cells

[HTML]

File Location and Name: C:\MyProjects\table\td.html

```
<html>
  <body>
    <table border = "1">
      <tr>
        <td >Sample</td>
        <td align="center">Text Aligned Cell</td>
      </tr>
      <tr>
        <td bgcolor="Blue">bgcolor cell</td>
        <td >Cell</td>
      </tr>
      <tr>
        <td colspan=2>Colspan cell</td>
      </tr>
      <tr>
        <td height="50">Custom height cell</td>
        <td nowrap>nowrap cell</td>
      </tr>
    </table>
```

```

        <td rowspan=2>Rowspan cell</td>
        <td valign="bottom" height="80">V align cell</td>
    </tr>
    <tr>
        <td width="300">custom width cell</td>
    </tr>
    <tr>
        <td>Cell</td>
        <td>Cell</td>
    </tr>
</table>
</body>
</html>

```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\table\td.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\table\td.html")
```

4.6.30 TEXTAREA - Text Area Tag

The **Text Area** tag is used to define a multiline text box. The HTMLUI control supports the following attributes with the **<textarea>** tag for receiving inputs from the user effectively.

- **rows:** Specifies the number of rows for the text area
- **cols:** Specifies the number of columns for the text area

[HTML]

File Location and Name: C:\MyProjects\textArea\textarea.html

```

<html>
  <body>
    <textarea rows="5" cols="20">
      Essential Studio features "Just-In-Time" source level debugging. Switch
      between debug and release versions with a single click.
    </textarea>
  </body>
</html>

```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\textArea\textarea.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\textArea\textarea.html")
```

4.6.31 TH - Table Header Tag

The **Table Header** tag is used to define header cells for the cells in a table. The **<th>** tag renders the text content in the particular cell in a bold face. The **<th>** tag supports the following attributes:

- **align**: Specifies the alignment of the text inside the table cell
- **bgcolor**: Specifies a background color for the specified cell
- **colspan**: Spans the cell to the specified number of columns. This is used in merging the columns in the table.
- **height**: Specifies custom height for the cells
- **nowrap**: Extends the text inside a particular cell into a single line. This display extends the width of the cell according to the contents inside it.
- **rowspan**: Extends the height of the cell to the specified number of rows. This is helpful in custom merging the rows of the given cell.
- **valign**: Determines the vertical alignment of the text inside the table cell
- **width**: Specifies user-defined width for the specified cells

[HTML]

```
File Location and Name: C:\MyProjects\table\th.html
```

```
<html>
  <body>
    <table border = "1">
      <tr>
        <th colspan = 2>Essential Studio</th>
      </tr>
      <tr>
        <td>Essential</td>
        <td>Tools</td>
      </tr>
      <tr>
        <td>Essential</td>
        <td>Grid</td>
      </tr>
      <tr>
        <td>Essential</td>
        <td>Chart</td>
      </tr>
      <tr>
        <td>Essential</td>
        <td>HTMLUI</td>
      </tr>
    </table>
  </body>
</html>
```

```
        </tr>
    </table>
</body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\table\th.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\table\th.html")
```

4.6.32 TITLE - Title Tag

The **Title** tag is used to enter a title to the document. The **<title>** tag is displayed in the title bar of the HTMLUI control, which is present at the top of the control. The title display can be toggled with the help of **ShowTitle** property of the HTMLUI control. This is a bool property, which when set to true displays the title bar over the control. The title tag is inserted in the head section of the document.

[HTML]

File Location and Name: C:\MyProjects\head\title.html

```
<html>
  <head>
    <title>
      Syncfusion Essential HTMLUI
    </title>
  </head>
  <body>
    <p>100% .NET HTML display engine that can be used to create extremely
    flexible user interfaces. Part of Essential Studio Enterprise.</p>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\head\title.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\head\title.html")
```

4.6.33 TR - Table Row Tag

The **Table Row** tag is used to define a row in a table. The row contains the cells (td elements) as its children in a table. HTMLUI supports the **align** attribute that aligns the contents of the row to the specified horizontal alignment.

[HTML]

File Location and Name: C:\MyProjects\table\tr.html

```
<html>
  <body>
    <table border = "1">
      <tr align="center">
        <td>Syncfusion</td>
        <td>Essential Studio</td>
      </tr>
    </table>
  </body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\table\tr.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\table\tr.html")
```

4.6.34 UL - UnOrdered List Tag

The **UnOrdered List** tag defines the start of a bulleted list. HTMLUI supports the bullets of the following shapes: **Circle**, **Square** and **Disc**. The unordered list contains the **** items as its child elements. Each **** element define a list item. The **type** attribute is used to choose the required bulleting style.

[HTML]

File Location and Name: C:\MyProjects\listItem\ul.html

```
<html>
  <body>
    <ul type="disc">
      <li>Essential Tools</li>
      <li>Essential Chart</li>
    </ul>
  </body>
</html>
```



```
<li>Essential HTMLUI</li>
</ul>
</body>
</html>
```

[C#]

```
this.htmluiControl.LoadHTML(@"C:\MyProjects\listItem\ul.html");
```

[VB.NET]

```
Me.htmluiControl.LoadHTML(@"C:\MyProjects\listItem\ul.html")
```

4.6.35 HTML Tags Sample

This sample shows the various Tags supported in HTMLUI.

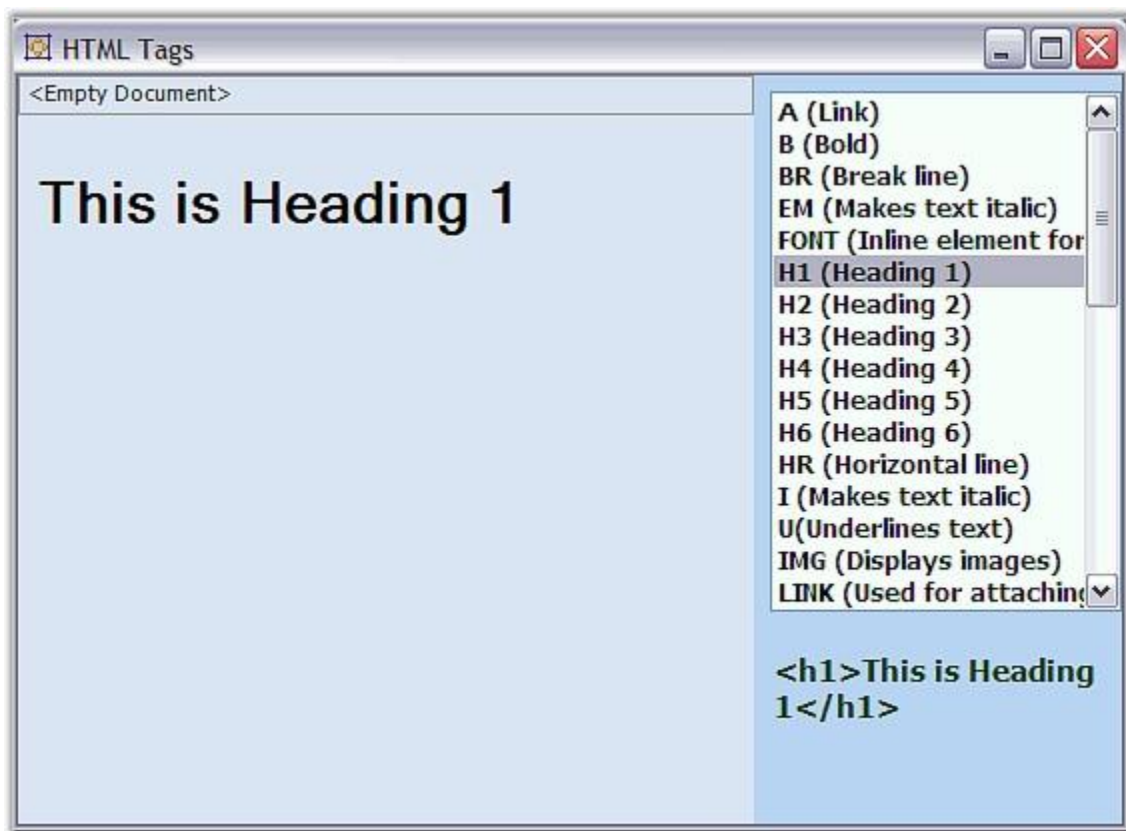


Figure 27: HTMLTags Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\ Samples\2.0\HTML Tags\HTMLTags

4.7 Custom Controls

The Custom Controls are not standard HTML elements but user-defined controls that are created for improving the application's richness and productivity.

The **Custom** tag is used to include the custom controls in an HTML document. The custom tag comes with two attributes: **assembly** and **class**.

The **assembly** attribute refers to the namespace where the control is located. The **class** attribute represents the control.

An HTML document containing custom controls is shown below.

```
[HTML]

<html>
  <body>
    <div>
      CheckBoxAdv:<CUSTOM class="Syncfusion.Windows.Forms.Tools.CheckBoxAdv"
        assembly="Syncfusion.tools.windows">
    </CUSTOM>
    </div>
    <div>
      NumericUpDown:<CUSTOM class="NumericUpDown" assembly="System.Windows.Forms"></CUSTOM>
    </div>
  </body>
</html>
```

The custom controls defined in the HTML document are interfaced with their equivalent windows forms control with the help of the **PreRenderDocument** event. The PreRenderDocument event occurs at a time when the HTML document is being loaded into the HTMLUI control, but the elements are not yet positioned.

The HTML elements are loaded into an hash table with an equivalent id as their key. An equivalent Base class object, here **BaseElement** class, is defined to link the HTML elements stored in the hash table with the help of the key associated with the element. The BaseElement is the Base class for all HTML elements. All HTML tag elements inherit this class.

The **CustomControlBase** implements the base functionality of the Windows forms control on the HTML tag element.

```
[C#]
```

```
private void htmluiControl1_PreRenderDocument(object sender,
Syncfusion.Windows.Forms.HTMLUI.PreRenderDocumentArgs e)
{
    Hashtable htmlelements = new Hashtable();
    htmlelements = e.Document.ElementsByUserID;

    // Here the base functionality of the 'this.checkBoxAdv1' is implemented to the
    'checkBoxAdvElement1'.
    BaseElement CheckBoxAdvElement1 = htmlelements["CheckBoxAdv"] as BaseElement;

    // Create a new Wrapper object.
    new CustomControlBase( CheckBoxAdvElement1, this.CheckBoxAdv1 );
    BaseElement NumericUpDownElement = htmlelements["NumericUpDown"] as BaseElement;
    new CustomControlBase( NumericUpDownElement, this.NumericUpDown1 );
}
```

[VB.NET]

```
Private Sub htmluiControl1_PreRenderDocument(ByVal sender As Object, ByVal e As
Syncfusion.Windows.Forms.HTMLUI.PreRenderDocumentArgs)
    Dim htmlelements As Hashtable = New Hashtable()
    htmlelements = e.Document.ElementsByUserID

    ' Here the base functionality of the 'this.checkBoxAdv1' is implemented to the
    'checkBoxAdvElement1'.
    Dim CheckBoxAdvElement1 As BaseElement = CType(If(OfType htmlelements("CheckBoxAdv") Is
BaseElement, htmlelements("CheckBoxAdv"), Nothing), BaseElement)

    ' Create a new Wrapper object.
    Dim oTemp1 As CustomControlBase = New CustomControlBase(CheckBoxAdvElement1, Me.CheckBoxAdv1)
    Dim NumericUpDownElement As BaseElement = CType(If(OfType htmlelements("NumericUpDown") Is
BaseElement, htmlelements("NumericUpDown"), Nothing), BaseElement)
    Dim oTemp2 As CustomControlBase = New CustomControlBase(NumericUpDownElement, Me.NumericUpDown1)
End Sub
```

The following image illustrates three custom controls created using HTMLUI.

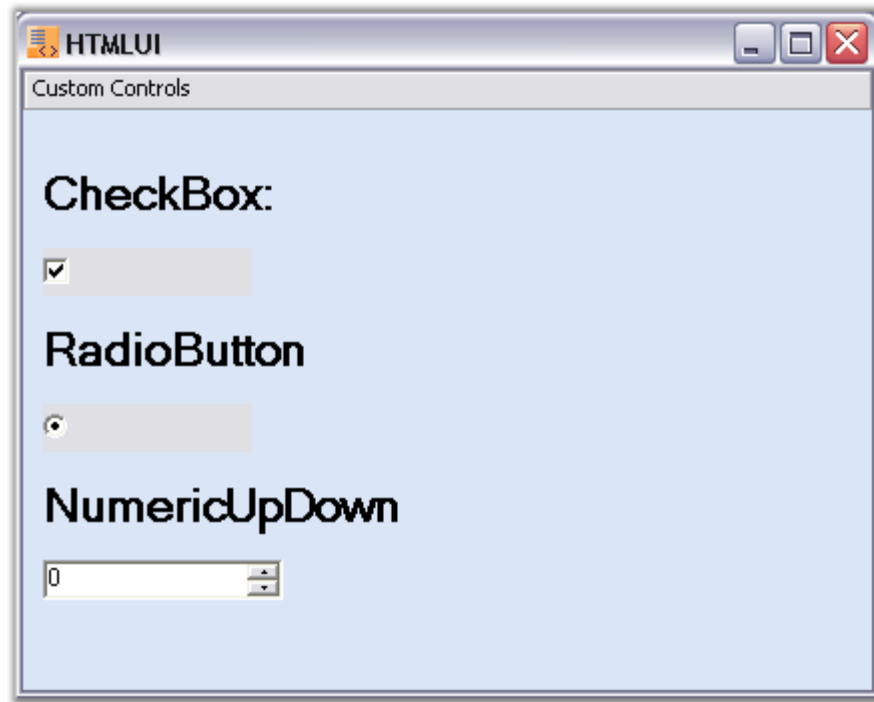


Figure 28: Custom Controls created by using the HTMLUI Control

4.7.1 Custom Controls Sample

This sample demonstrates the implementation of Custom Controls by using HTMLUI.

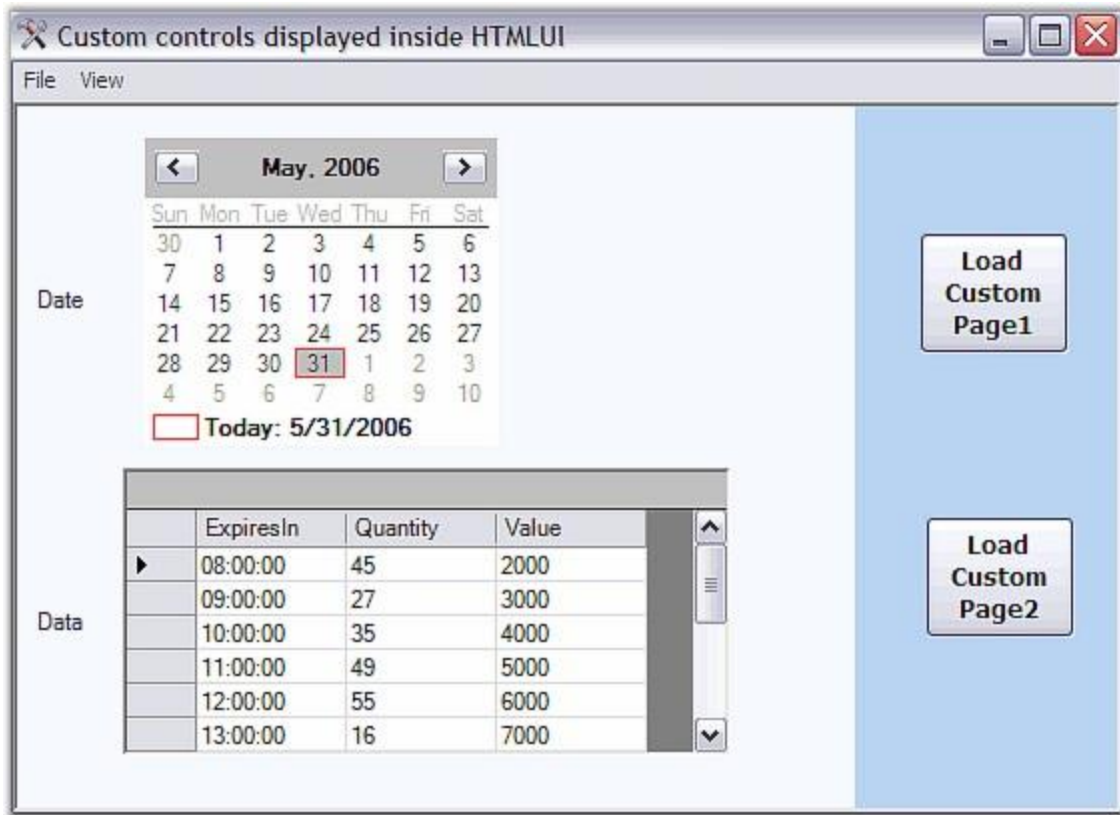


Figure 29: HTMLUICustomControls Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0\Custom Controls\HTMLUICustomControls

4.8 HTML Forms

Forms are the containers for placing the elements in a HTML document. HTMLUI supports the usage of forms for developing advanced and decorative pages in the user's application.

```
[HTML]

<html>
  <body>
    <form>
      <input type = "text"/><br/>
      <input type = "button"/><br/>
      <input type = "checkbox" checked /><br/>
    </form>
  </body>
</html>
```

Loading the above HTML document into HTMLUI creates a Form with the three controls specified as shown below.

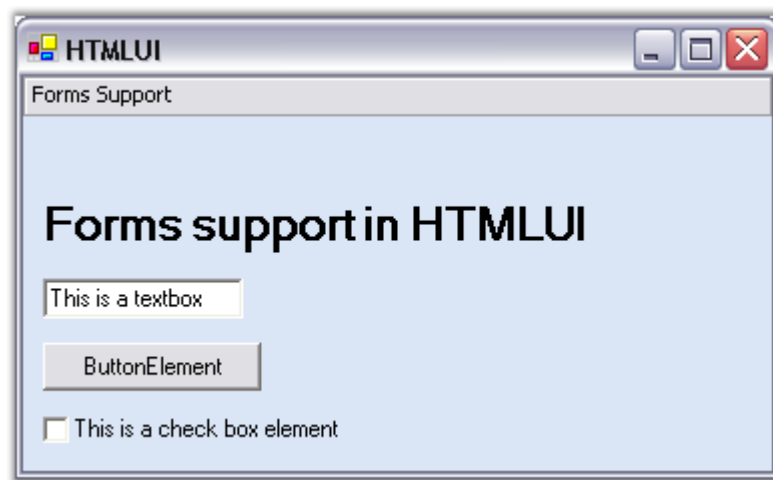


Figure 30: HTML Form support in the HTMLUI Control

4.8.1 HTMLUIForms Sample

This sample illustrates the implementation of Forms by using HTMLUI.

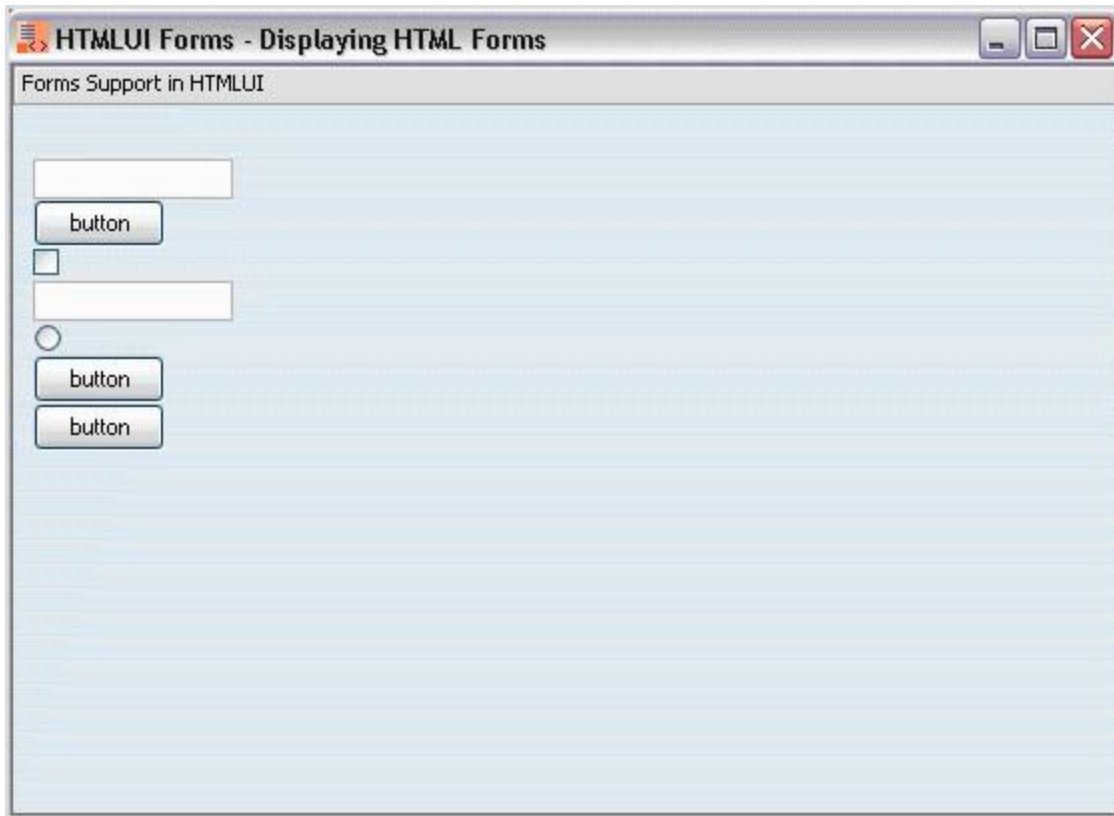


Figure 31: HTMLUIForms Sample

4.9 HTML Bookmarks

Bookmarks feature is enabled in the HTMLUI control. This allows the user to switch to particular references in the page when the link is clicked. The HTMLUI control has another functionality of referring bookmarks which is, referring them not only in the same page, but also in other pages.

```
[HTML]

<html>
  <body>
    <a href="Newfile.htm#bookmark"> New file Book mark </a>
    <a href="#bookmark"> Bookmark in same file </a>
    .....
    .....
    <div id="bookmark"> Syncfusion's HTMLUI Control </div>
  </body>
</html>
```

The following image shows the bookmarks implemented using HTMLUI.

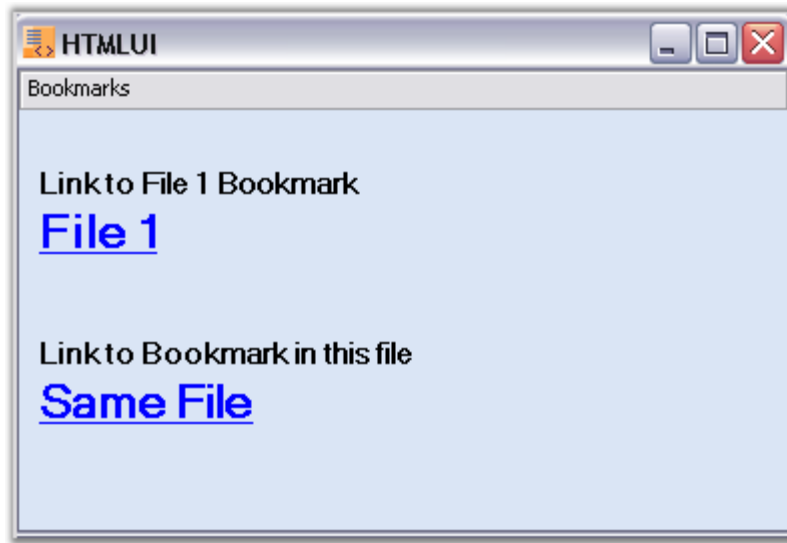


Figure 32: HTML Bookmarks inserted into the HTMLUI Control

4.9.1 HTMLUI Bookmarks Sample

This sample illustrates the implementation of Bookmarks in HTMLUI.

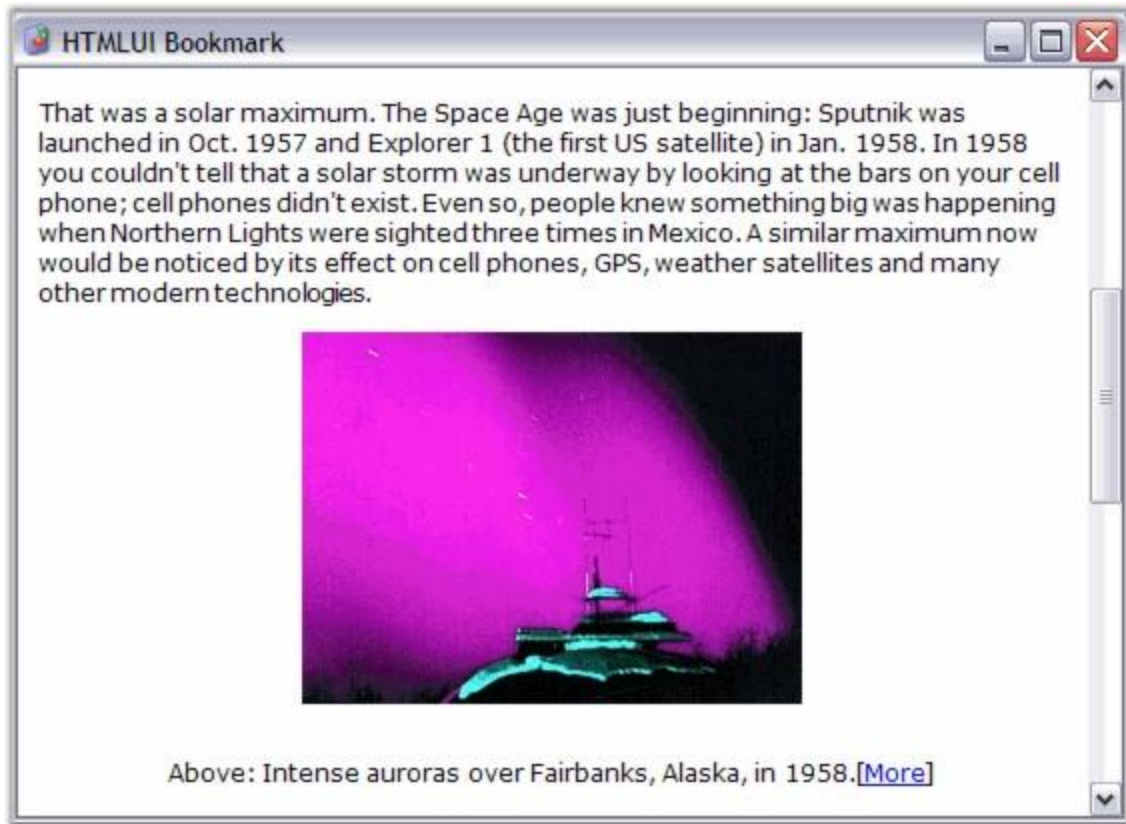


Figure 33: HTMLUIBookmarks Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0\HTML Tags\HTMLUIBookmarks

4.10 HTML Tables

HTMLUI has a rich table support that lets the user to decide the table's dimensions and design. The HTMLUI table support also comes with different alignments of text within the table. This helps the user in creating advanced and highly structured HTML applications.

[HTML]

```
<html>
  <body>
    <table>
      <tr><td>Table support in HTMLUI</td></tr>
    </table>
```

```
</body>
</html>
```

The HTML document that defines the Tables is then loaded into HTMLUI using any of the ways discussed in the section [Loading HTML](#).

4.10.1 HTMLUI Tables Sample

This sample illustrates how to implement Tables using HTMLUI.

HTMLUI Tables support

Tables Support in HTMLUI

Table 1

This is a column with Colspan=3		
Column 1	Column 2	Column 3
Column 1 with custom height	Column 2	Column 3

Table 2

Column 1 with rowspan=2	Column 2	Column 3
	Column 2	Column 3

Table 3

Column 1	Nested Table		
	Column 1	Column 2	Column 3
	Column 1	Column 2	Column 3

Figure 34: HTMLUITables Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0\HTML Tags\HTMLUITables

4.11 HTML Layout

HTMLUI can be used as a HTML viewer in two ways.

- Display Engine
- Layout Engine

The display purposes involve the functionality similar to Web browsers in displaying the contents of the HTML document. In the Layout purpose, it is used to layout and customize rich user-interactive interfaces.

HTMLUI control can be used in a variety of applications which are common in our day-to-day life.

With HTMLUI control's support to images and animated images, Chat applications can be developed. The form based dialog box applications used in the office desks can also be developed at ease by simply changing different HTML documents as per the needs. The other interesting applications that can be developed using HTMLUI include games, animations, user blogs, and so on.

The following figure shows a form based dialog that illustrates HTMLUI as a Layout Engine.



Figure 35: HTMLUI Control functioning as a Layout Engine

4.11.1 HTMLUI Chat Sample

This sample illustrates how a Chat application can be implemented using HTMLUI.



Figure 36: HTMLUIChat Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0\HTML Layout\HTMLUIChat

4.12 HTML Renderer

As the HTMLUI control supports rendering of web pages, it can be used like a light-weight web browser for compact applications that include links to references.

[C#]

```
// Load the specified HTML document from System.Uri in to HTMLUI Control and renders it.
string path = "http://www.Google.com";
Uri uri = new Uri(path);
htmluiControl1.LoadHTML(uri);
```

[VB.NET]

```
' Load the specified HTML document from System.Uri in to HTMLUI Control and renders it.
Private path As String = "http://www.Google.com"
Private uri As Uri = New Uri(path)
HtmluiControl1.LoadHTML(uri)
```

Also the ability of the HTMLUI control to load from strings can be used in creating HTML editors for tutorial applications.

[C#]

```
// Load HTML Document from String.
string htmlString = "<HTML>
<BODY> Document loaded through the LoadFromString method </BODY>
</HTML>";
this.htmluiControl1.LoadFromString(htmlString);
```

[VB.NET]

```
Private htmlString As String = "<HTML>
<BODY>Document loaded through the LoadFromString method</BODY>
</HTML>"
Me.HtmluiControl1.LoadFromString(htmlString)
```

The following figure shows an HTML Editor rendered using HTMLUI.

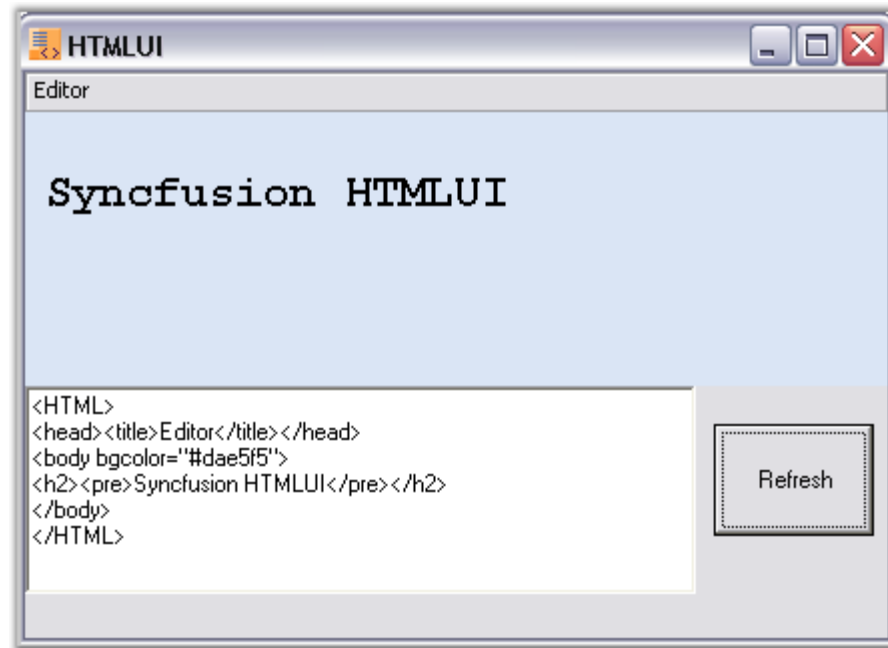


Figure 37: HTML Editor rendered by using the HTMLUI Control

4.12.1 HTMLUI Browser Sample

This sample demonstrates the implementation of a Web Browser in HTMLUI.



Figure 38: HTMLUIExplorer Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0\HTML Renderer\HTMLUIExplorer

4.12.2 HTMLUI Editor Sample

This sample demonstrates the implementation of HTML Editors in HTMLUI.

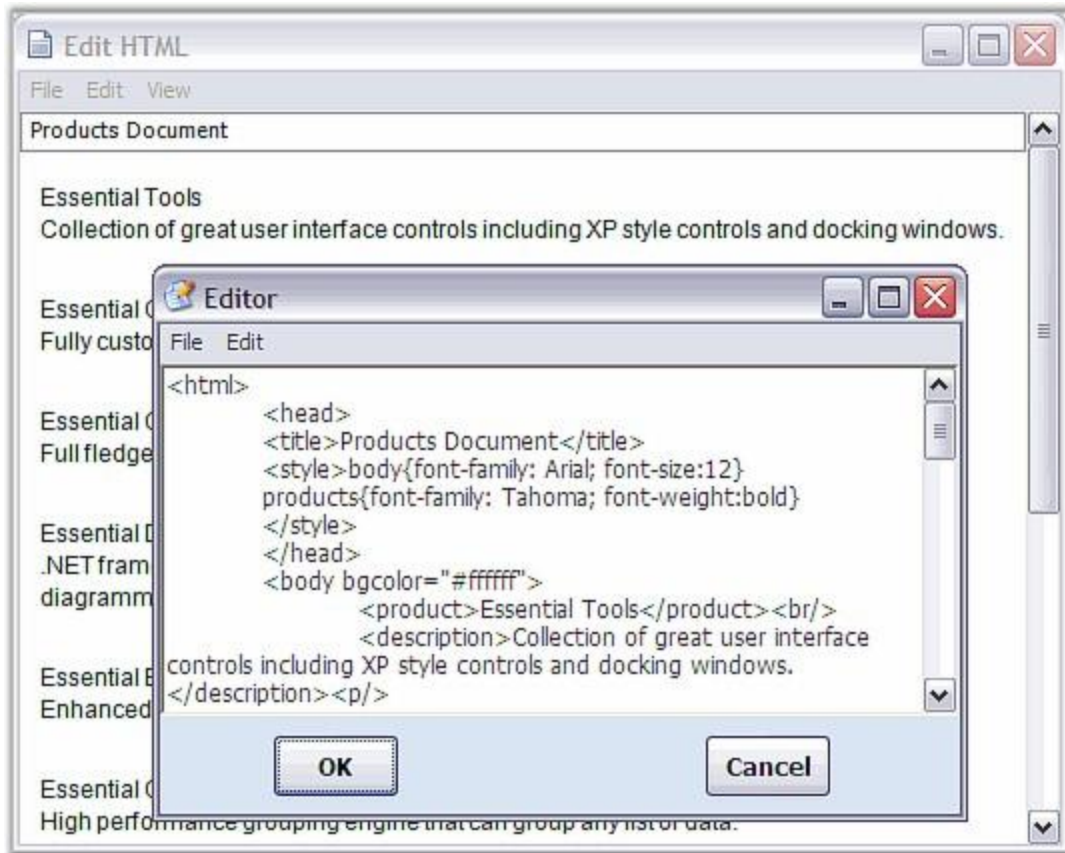


Figure 39: HTMLUIEditor Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0\HTML Renderer\HTMLUIEditor

4.13 Style Sheets CSS

The support for style sheets is enabled in HTMLUI. This lets the user to define styles for HTML elements and decide the appearance of the HTML elements in the application. HTMLUI supports three types of style sheets.

- External Style sheets
- Internal Style sheets
- Inline Style sheets

External style sheets are linked to the file through the **Link** tag. While the internal style sheets are applied with the help of the **Style** tag inside the head section, inline style sheets are applied as the values of the style attributes of the specific HTML element.

The following HTML code illustrates the above concepts.

```
[HTML]

<html>
  <head>
    <link type="text/css" rel="stylesheet" href="ExternalCSS.CSS"></link>
    <style>.div{"background-color: #dae5f5;"</style>
  </head>
  <body>
    <p style="background-color: #ffffff;"> This is an inline styled element </p>
    <div class="div"> The Internal style sheet is applied for this element </div>
  </body>
</html>
```

HTMLUI also supports updation of styles to the HTML document at run time. This can be done in two modes, either by changing the value of the style attribute for internal style or the class attribute for the inline style sheet.

```
[C#]

if(this.textBox.Attributes.Contains("style") == false)
this.textBox.Attributes.Add("style");
this.textBox.Attributes["style"].Value = "background-color:red;";
```

```
[VB.NET]

If Me.textBox.Attributes.Contains("style") = False Then
Me.textBox.Attributes.Add("style")
End If
Private Me.textBox.Attributes("style").Value = "background-color:red;"
```

4.13.1 Cascading Style Sheets

Cascading style sheets contain the style definitions for various HTML tags that are defined in the document. Style sheets are included in a HTML document to make the document more clearer so that the actual contents will be inside the document while the styles for these contents will be applied from an external resource. This is to improve the readability of the document and also to apply style changes to the existing document at different times when required.

HTMLUI supports the following style sheets for adding styles to the HTML document:

4.13.1.1 Inline StyleSheet

The Inline style sheet will be present inside the HTML tag as an attribute named **Style**, for the HTML element. The styles for the contents of the tag will be given as the values for this attribute.

The following snippet shows an inline style, which is applied as an attribute inside the tag to a HTML element, which is inside the document.

[HTML]

File name and location: C:\MyProjects\StyleSheets\inline.html

```
<html>
<body>
<p style="background-color: #dae5f5;">
Inline style applied to a paragraph.
</p>
</body>
</html>
```

[C#]

```
htmluiControl.LoadHTML(@"C:\MyProjects\StyleSheets\inline.html");
```

[VB.NET]

```
htmluiControl.LoadHTML(@"C:\MyProjects\StyleSheets\inline.html")
```

4.13.1.2 Internal StyleSheet

The Internal style sheet is used to define the same styles to all the occurrences of a specific tag in the document. The internal style sheet is defined inside the Style tag, in the head section of the document. The user can create another style definition for other HTML tags with a new name inside the same style tag. The following snippet shows how an internal stylesheet is defined for a specific tag in a HTML document.

[HTML]

File name and location: C:\MyProjects\StyleSheets\internal.html

```
<html>
<head>
<style type="text/css">
    p {color: blue}
    div{color: red}
</style>
</head>
```

```
<body>
<p>This is a paragraph.</p>
<div>This is a division.</div>
<p>This is a new paragraph.</p>
</body>
</html>
```

[C#]

```
htmluiControl.LoadHTML(@"C:\MyProjects\StyleSheets\internal.html");
```

[VB.NET]

```
htmluiControl.LoadHTML(@"C:\MyProjects\StyleSheets\internal.html")
```

4.13.1.3 External StyleSheet

The External style sheets contain style definitions in a separate .css file, for various HTML tags that are in the document. These styles are applied by linking the css file to the HTML document inside the Link tag. The Link tag should be placed in the head section of the HTML document as it contains the information about the cascading style sheet that is to be referred by this document.

[CSS]

FileName and location: C:\MyProjects\StyleSheets\styleSheet.css

```
body
{
    background-color: #dae5f5;
    cursor: default;
}
p
{
    color: Green;
}
div
{
    color: Blue;
    font-family: Tahoma;
}
```

[HTML]

File name and location: C:\MyProjects\StyleSheets\external.html

```
<html>
<head>
    <link rel=Stylesheet type="text/css"
href="C:\MyProjects\StyleSheets\styleSheet.css" />
</head>
<body>
<p>Green color for paragraph.</p>
<div>Blue color for division</div>
</body>
</html>
```

The HTMLUI control uses two modes of applying styles to the HTML document with the help of the external style sheets.

4.13.1.3.1 Design Time

This type of setting is carried out in the document at design time. It is used in circumstances where only a standard style is applied for the documents.

[C#]

```
htmluiControl.LoadHTML(@"C:\MyProjects\StyleSheets\external.html");
```

[VB.NET]

```
htmluiControl.LoadHTML(@"C:\MyProjects\StyleSheets\external.html")
```

4.13.1.3.2 Run Time

HTMLUI is so flexible that the user can define styles for the HTML document at run time. The **LoadCSS** method of the HTMLUI control helps the user to load another CSS file to the current document at run time.

[C#]

```
htmluiControl.LoadHTML(@"C:\MyProjects\StyleSheets\external.html");
htmluiControl.LoadCSS(@"C:\MyProjects\StyleSheets\NewStyleSheet.css");
```

[VB.NET]

```
htmluiControl.LoadHTML(@"C:\MyProjects\StyleSheets\external.html"
htmluiControl.LoadCSS(@"C:\MyProjects\StyleSheets\NewStyleSheet.css")
```


4.13.1.4 Class Selectors

Internal and external styles are not only defined by mentioning the names of the tags, but also by the **Style Class Selectors**. The class selectors are used to define the styles under a common class name and apply the styles to different tags by specifying the name of the class as the value of the **class** attribute, for the specific element.

HTMLUI supports two types of styles definitions for the HTML documents with the help of the class selectors as given below:

4.13.1.4.1 Name Class Selectors

The Name Class Selectors contain a common name that is given to the style class, which is defined in the internal or external css file. The following snippet shows how the name class selectors are defined for html elements in the document.

[HTML]

File name and location: C:\MyProjects\StyleSheets\NameClass.html

```
<html>
<head>
  <style>
    .red {color: red}
    .blue {color: blue}
  </style>
</head>
<body>
  <h1 class="red">Red Heading</h1>
  <p class="blue">Blue colored paragraph.</p>
</body>
</html>
```

[C#]

```
htmluiControl.LoadHTML(@"C:\MyProjects\StyleSheets\NameClass.html");
```

[VB.NET]

```
htmluiControl.LoadHTML(@"C:\MyProjects\StyleSheets\NameClass.html")
```

4.13.1.4.2 ID Class Selectors

The ID Class Selectors are defined like the name class selectors. Instead of specifying names, a unique identity is specified for the styles while defining them, and these style names are passed as the values of the **id** attribute to the concerned HTML elements.

[HTML]

File name and location: C:\MyProjects\StyleSheets\idClass.html

```
<html>
<head>
  <style>
    #red {color: red}
    #blue {color: blue}
  </style>
</head>
<body>
  <h1 id="red">Red Heading</h1>
  <p id="blue">Blue colored paragraph.</p>
</body>
</html>
```

[C#]

```
htmluiControl.LoadHTML(@"C:\MyProjects\StyleSheets\idClass.html");
```

[VB.NET]

```
htmluiControl.LoadHTML(@"C:\MyProjects\StyleSheets\idClass.html")
```

4.13.1.5 CSS Comments

The HTMLUI control supports comments inside the cascading style sheet. The comments helps the developer to explain his ideas behind the styles and also help the user to understand the functionality of the styles. A CSS comment begins with `/*` and ends with `*/` (For eg., `/*This is a comment*/`). The comments are not visible in the browser at run time.

[CSS]

```
/*blue colored font for the p elements*/
p {color: blue}

/*red colored font for the div elements*/
div{color: red}
```

4.13.2 Supported CSS Attributes

The following attributes are supported in HTMLUI for the cascading style sheet definition.

4.13.2.1 Background - CSS

The **background** attribute of CSS helps the user to set the back ground properties for the specified element. The following are the background properties that are supported in HTMLUI for a HTML element using CSS.

4.13.2.1.1 Background Image

The **background-image** property is used to set an image as the background of the HTML element.

```
[HTML]

<html>
  <head>
    <style type="text/css">
      /*The url attribute gets the image from the specified location*/
      body{background-image: url(stars.gif);}
    </style>
  </head>
  <body>
    <p>Image in the background.</p>
  </body>
</html>
```

4.13.2.1.2 Background Color

The **background-color** property is used to set a background color for a HTML element.

```
[HTML]

<html>
<head>
<style type="text/css">
/*A background color is specified for the document*/
  body{background-color: #dae5f5;}
</style>
</head>
<body>
```



```
<p>Back ground color set to the body element.</p>
</body>
</html>
```

4.13.2.1.3 Background Repeat

The **background-repeat** property repeats the background that is set for the HTML element, horizontally or vertically.

- **repeat-x**: Repeats the background image horizontally
- **repeat-y**: Repeats the background image vertically
- **repeat**: Repeats the background image both horizontally and vertically
- **no-repeat**: Fixes the background image at the top-left corner of the element and does not repeat the image display

```
[HTML]

<html>
<head>
<style type="text/css">
/*A background image is repeated horizontally.*/
body{background-image: url(sync.gif); background-repeat: repeat-x;}
</style>
</head>
<body>
<p>Back ground image for the document repeated horizontally.</p>
</body>
</html>
```

4.13.2.2 Text – CSS

The **text** attribute in CSS helps the user to set properties for his text display. With these properties the appearance of the text can be modified according to the users requirements.

4.13.2.2.1 Text Color

The **color** attribute is used to apply a specific color to the rendered text.

```
[HTML]

<html><head>
<style type="text/css">
h1 {color: #00ff00}
```

```
p {color: blue}
</style>
</head>
<body>
<h1>Text - HTMLUI - CSS</h1>
<p>This is a paragraph</p>
</body>
</html>
```

4.13.2.2.2 Text Align

The **text-align** attribute is used to align the text inside an element to the specified horizontal direction.

```
[HTML]

<html>
<head>
<style type="text/css">p{ text-align: center}</style>
</head>
<body>
<p>Center aligned paragraph</p>
</body>
</html>
```

4.13.2.2.3 Text Decoration

The **text-decoration** attribute is used to decorate a text. The HTMLUI control supports the **Underline** and the **None** style values for the text decoration attribute.

```
[HTML]

<html><head>
<style type="text/css">.five:hover {text-decoration: underline} </style>
</head>
<body>
<p>Mouse over the links to see them change layout.</p>
</body></html>
```

4.13.2.3 Font – CSS

The **font** attribute is used to define the font specification for a HTML element in the HTMLUI. With the CSS font specification, the user can create good presentation applications in HTMLUI.

The HTMLUI control supports the following font attributes for the HTML elements.

4.13.2.3.1 Font Family

The **font-family** attribute is used to specify a font for the text that is to be displayed in the specific HTML element.

[HTML]

```
<html><head>
<style>p {font-family: Courier New}</style>
</head>
<body>
<p>This paragraph is given a font through styles.</p>
</body>
</html>
```

4.13.2.3.2 Font Size

The **font-size** attribute is used to specify a size for the rendered text that is to be displayed in the HTMLUI control.

[HTML]

```
<html><head>
<style>p {font-size: 15}</style>
</head>
<body>
<p>A font size applied to the paragraph element.</p>
</body>
</html>
```

4.13.2.3.3 Font Style

The **font-style** attribute is used to format the specified text with the given styles. HTMLUI supports the **normal** and **italic** styles for the rendered text inside the HTML element.

[HTML]

```
<html><head>
<style>p {font-style: italic}</style>
</head>
<body>
<p>Italic style applied to the formatted text.</p>
```

```
</body>
</html>
```

4.13.2.3.4 Font Weight

The **font-weight** attribute is used to specify the thickness or boldness of the rendered text. HTMLUI supports the **normal** and **bold** font-weight that is associated with the CSS text rendering.

```
[HTML]

<html><head>
<style>p{font-weight: bold}</style>
</head>
<body>
<p>Bold font applied to paragraph.</p>
</body></html>
```

4.13.2.4 Border – CSS

The **border** attribute is used to specify the border properties for the rendered table element through CSS. The color and thickness of the border can also be specified through the border attribute. HTMLUI also supports applying border properties to the four sides of the table border independently.

4.13.2.4.1 Border Color

- **border-color**: Specifies the color for the border of the rendered table
- **border-bottom-color**: Specifies the color for the bottom border of the rendered table
- **border-top-color**: Specifies the color for the top border of the rendered table
- **border-left-color**: Specifies the color for the left border of the rendered table
- **border-right-color**: Specifies the color for the right border of the rendered table

```
[HTML]

<html><head>
<style>
.table { border-color: blue }
table{ border-left-color: blue; border-right-color: green; border-top-color:
red; border-bottom-color: black }
</style>
</head>
<body>
```

```
<table border=1><tr><td>This is a table.</td></tr></table><br/>
<table class="table" border=1><tr><td>This is another table.</td></tr></table>
</body></html>
```

4.13.2.4.2 Border Width

- **border-width:** Specifies the thickness for the border of the rendered table. The user can specify the border-width in units.
- **border-bottom-width:** Specifies the thickness for the bottom border of the rendered table
- **border-top-width:** Specifies the thickness for the top border of the rendered table
- **border-left-width:** Specifies the thickness for the left border of the rendered table
- **border-right-width:** Specifies the thickness for the right border of the rendered table

[HTML]

```
<html><head>
<style>
.table { border-width: 5 }
table{border-left-width: 3; border-right-width: 5; border-top-width: 6;
border-bottom-width: 8}
</style>
</head>
<body>
<table border=1><tr><td>This is a table.</td></tr></table><br/>
<table class="table" border=1><tr><td>This is another table.</td></tr></table>
</body></html>
```

4.13.2.5 Padding – CSS

The CSS **padding** attribute in HTMLUI is used to define a fixed space between the element's border and its contents. The top, right, bottom and left padding attributes can be specified independently.

[HTML]

```
<html><head>
<style>
.table { padding: 5 }
table{padding-left: 25; padding-right: 50; padding-top: 25; padding-bottom: 50}
</style>
</head>
<body>
<table border=1><tr><td>This is a table.</td></tr></table><br/>
<table class="table" border=1><tr><td>This is another table.</td></tr></table>
```

```
</body>  
</html>
```

4.13.2.5.1 Padding

The **padding** attribute is used to specify a fixed space for all the four sides of the element's contents.

4.13.2.5.2 Padding Right

The **padding-right** attribute is used to specify the padding property at the right side of the HTML element.

4.13.2.5.3 Padding Left

The **padding-left** attribute is used to specify the padding for the left side of the element.

4.13.2.5.4 Padding Top

The **padding-top** attribute specifies the padding on the top edge of the element.

4.13.2.5.5 Padding Bottom

The **padding-bottom** attribute specifies the padding at the bottom of the HTML element from its border.

4.13.2.6 Dimension - CSS

The Dimension properties are used to specify the size for an HTML element.

- **height:** Specifies the height of an element. The HTMLUI control allows the user to specify the height either in percentage or units.
- **width:** Specifies the width of an element

The width of an element in HTMLUI can be mentioned in terms of units.

```
[HTML]
```

```
<html>
<head>
<style type="text/css">p { height: 25%; width:200;} table{height:
100%;}</style>
</head>
<body>
<table border=1><tr><td><p>NewSample</td></tr></table>
</body>
</html>
```

4.13.2.7 Classification - CSS

The CSS Classification properties determine the display of an element in the HTMLUI control. The HTMLUI control supports the following classification property that determines how and where to display a HTML element.

- **cursor:** Specifies the cursor that is to be displayed when the mouse pointer is over the specified HTML element

[HTML]

```
<html>
<head><style>p{cursor: cross}</style></head>
<body>
<p>The HTMLUI Editor helps you to edit your HTML document.</p>
</body>
</html>
```

4.13.2.8 Positioning - CSS

The **positioning** attribute is used to determine the position of the HTML element that is to be displayed in the HTMLUI control.

- **vertical-align:** Specifies the vertical alignment of the element during display

[HTML]

```
<html>
<head>
<style type="text/css">td{vertical-align: bottom}</style></head>
<body>
<table border=1 height="100%"><tr><td>Bottom Aligned table
cell.</td></tr></table>
</body></html>
```

4.13.2.9 Pseudo - Classes

The Pseudo-Classes are used to add special effects to the HTML hyperlink element display. For understanding this case clearly in HTMLUI, let us define a general case, where the link should change its appearance when a mouse pointer is moved over it.

The HTMLUI control supports the link and the hover classes in order to display the links in the HTMLUI control.

- **link:** Specifies the properties for the display of a hyperlink in the normal state
- **hover:** Specifies the CSS properties for the hyperlink when a mouse is moved over the link

The following sample shows how different properties can be set for the hyperlink element by using the pseudo-classes.

[HTML]

```
<html><head>
<style type="text/css">
.ChangeColor:link {color: #ff0000}
.ChangeColor:hover {color: #ffcc00}

.ChangeFont:link {color: #ff0000}
.ChangeFont:hover {font-size: 150%; font-family: Tahoma}

.ChangeBgcolor:link {color: #ff0000}
.ChangeBgcolor:hover {background-color: #66ff66}

.ChangeTextDec:link {color: #ff0000; text-decoration: none}
.ChangeTextDec:hover {text-decoration: underline}
</style></head>
<body>
<a class="ChangeColor" href="none.htm" target="_blank">This link changes
color</a><br/>
<a class="ChangeFont" href="none.htm" target="_blank">This link changes
font</a><br/>
<a class="ChangeBgcolor" href="none.htm" target="_blank">This link changes
background-color</a><br/>
<a class="ChangeTextDec" href="none.htm" target="_blank">This link changes
text-decoration</a><br/>
</body>
</html>
```

4.13.3 HTMLUIUseCSS Sample

This sample demonstrates the implementation of External Style Sheets in HTMLUI.

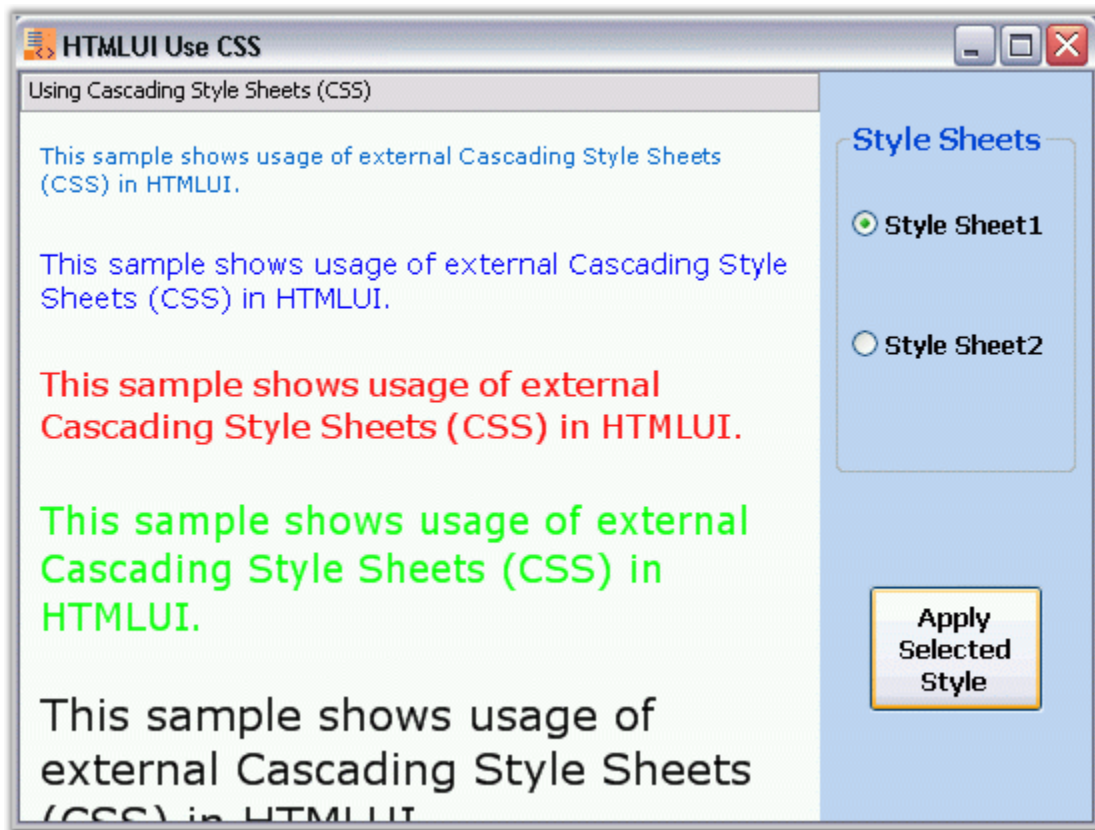


Figure 40: HTMLUIUseCSS Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0\HTMLUI Appearance\HTMLUIUseCSS

4.13.3.1 HTMLUIElementsCSS Sample

This sample demonstrates the implementation of Internal Style Sheets on HTML Elements.

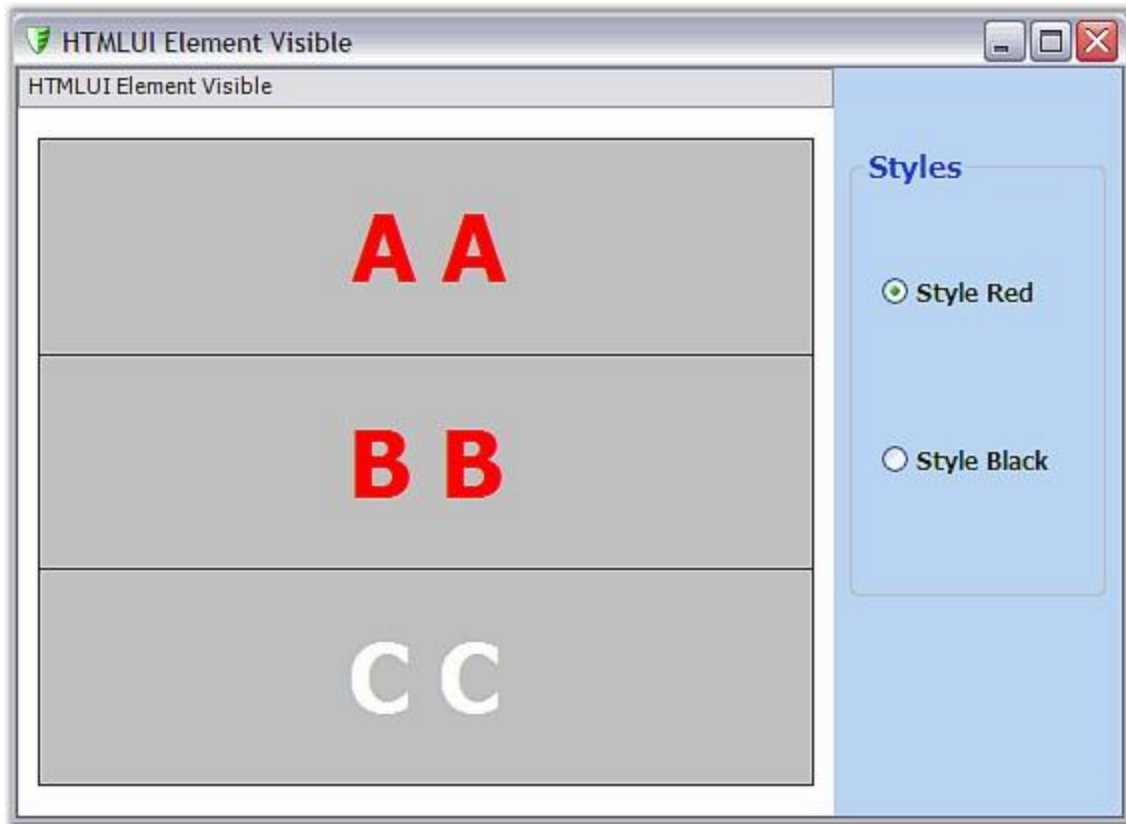


Figure 41: HTMLUIElementCSS Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0\HTMLUI Appearance\HTMLUIElementCSS

4.14 HTMLUI Appearance

With HTMLUI, the application developer can decide the appearance of his application even at run time by setting the style according to his needs. A common application is changing the background color for a particular page based on the user signing in. These interactive applications help in developing user-friendly applications.

The HTMLUI control allows the following modes of changing the background color to your application.

- Inside the HTML document as Attribute
- Using the Style Sheets

The following HTML coding shows the different methods of changing the background color that HTMLUI control supports, leaving it to the user to choose the best among the various options as per the requirements.

```
[HTML]

<html>
  <head>
    <link type="text/css" rel="stylesheet" href="backgroundColor.CSS"></link>
    <style>.div{"background-color: #ffffcc;"</style>
  </head>
  <body bgcolor="#dae5f5">
    <p style="background-color: #ffffff;">Background color changed through the style attribute</p>
  </body>
</html>
```

The following figure shows the bgcolor of the HTML document customized by using HTMLUI.



Figure 42: Background Color of the HTML document customized by using the HTMLUI Control

4.14.1 HTMLUIAppearance Sample

This sample illustrates the customization of HTMLUI Appearance.

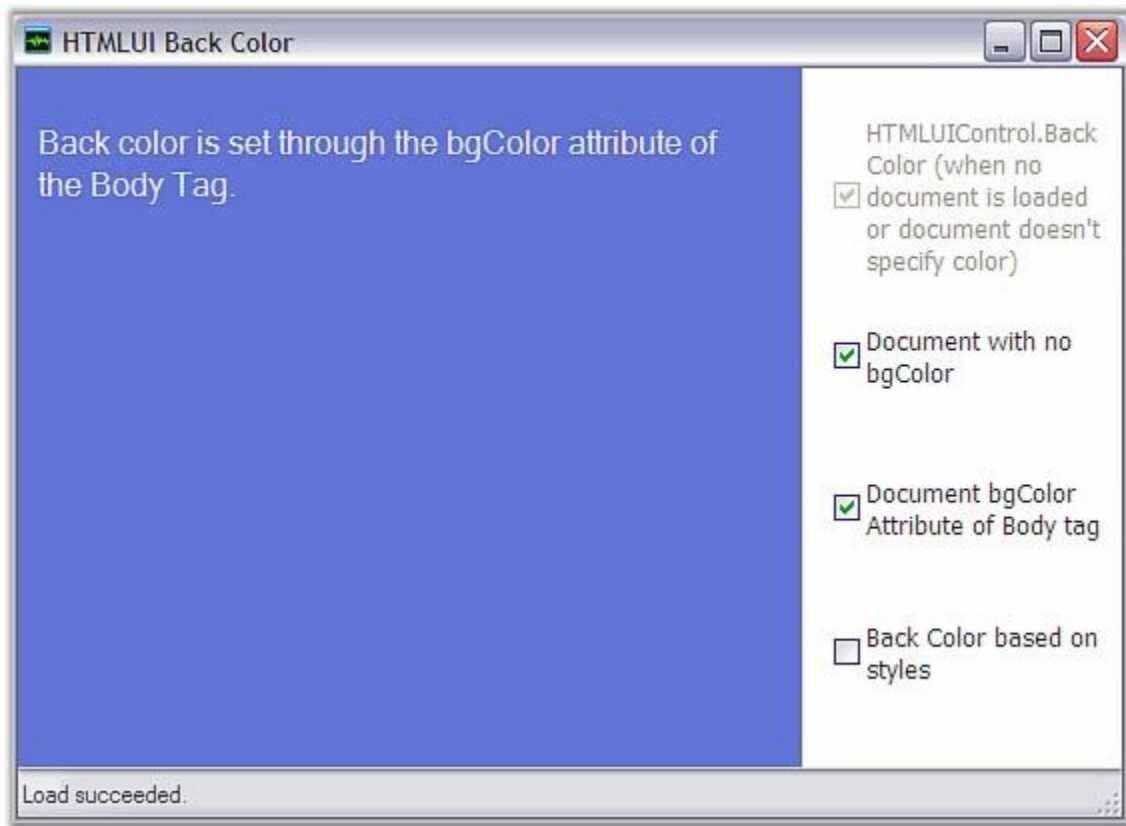


Figure 43: HTMLUIBackColor Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0\HTMLUI Appearance\HTMLUIBackColor

4.15 HTML Format

HTMLUI allows the user to apply formats to the elements at run time. The **HTMLFormat** class creates a format for the HTML elements displayed in the HTMLUI control. The user can apply the format on the execution of some events based on the necessity of the application.

```
[C#]

// Implementation of HTMLFormat interface
Hashtable html = this.htmluiControl1.Document.GetElementsByUserIdHash();
BaseElement div = html["div1"] as BaseElement;

HTMLFormat format = new HTMLFormat("ClickedPara");
```

```
format.Font = new Font("Verdana", 12);  
format.ForeColor = Color.Blue;  
div.Format = format;
```

[VB.NET]

```
' Implementation of HTMLFormat interface  
Private html As Hashtable = Me.HtmluiControl1.Document.GetElementsByIdHash()  
Private div As BaseElement = CType(If(OfType(Of html)("div1") Is BaseElement,  
html("div1"), Nothing), BaseElement)  
  
Private format As HTMLFormat = New HTMLFormat("ClickedPara")  
Private format.Font = New Font("Verdana", 12)  
Private format.ForeColor = Color.Blue  
Private div.Format = format
```

With HTMLUI, the user can also access the location of the elements in the HTMLUI control.

[C#]

```
Hashtable html = this.htmluiControl1.Document.GetElementsByIdHash();  
BaseElement div = html["element"] as BaseElement;  
beginPoint = element.Location;  
endPoint = new Point(beginPoint.X + element.Size.Width, beginPoint.Y +  
element.Size.Height);
```

[VB.NET]

```
Private html As Hashtable = Me.HtmluiControl1.Document.GetElementsByIdHash()  
Private div As BaseElement = CType(If(OfType(Of html)("element") Is BaseElement,  
Nothing), BaseElement)  
Private beginPoint = element.Location  
Private endPoint = New Point(beginPoint.X + element.Size.Width, beginPoint.Y +  
element.Size.Height)
```

4.15.1 HTMLFormat Sample

This sample shows how the styles are applied by using the HTML Format object.



Figure 44: HTMLFormat Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0\HTMLUI Format\HTMLFormatSample

4.16 Element Format

Essential HTMLUI supports formatting of not only the entire HTML document as a whole, but also the individual elements. With HTMLUI, the user can replace any HTML element into some other format before displaying, in a view to develop advanced user interactivity.

The following snippet shows how a text content can be replaced with an image in a text sequence.

[C#]

```
private const string DEF_IMG_TAG = "<img src='../..\clock.jpg'>";
private const string DEF_TIME = "time";
```

```
private void htmluiControl1_LoadFinished(object sender, System.EventArgs e)
{
    // Returns the html element by its ID, defined in HTML document.
    IHTMLElement p = this.htmluiControl1.Document.GetElementById("p");

    // Replace the HTML inner text of current element.
    p.InnerHTML = p.InnerHTML.Replace(DEF_TIME, DEF_IMG_TAG);
    this.htmluiControl1.Refresh();
}
```

[VB.NET]

```
Private Const DEF_IMG_TAG As String = "<img src='../..\clock.jpg'>"
Private Const DEF_TIME As String = "time"

Private Sub htmluiControl1_LoadFinished(ByVal sender As Object, ByVal e As
System.EventArgs)

    ' Returns the html element by its ID, defined in HTML document.
    Dim p As IHTMLElement = Me.htmluiControl1.Document.GetElementById("p")

    ' Replace the HTML inner text of current element
    p.InnerHTML = p.InnerHTML.Replace(DEF_TIME, DEF_IMG_TAG)
    Me.htmluiControl1.Refresh()
End Sub
```

The following image shows the text element **Time** replaced by an image while displayed using HTMLUI.

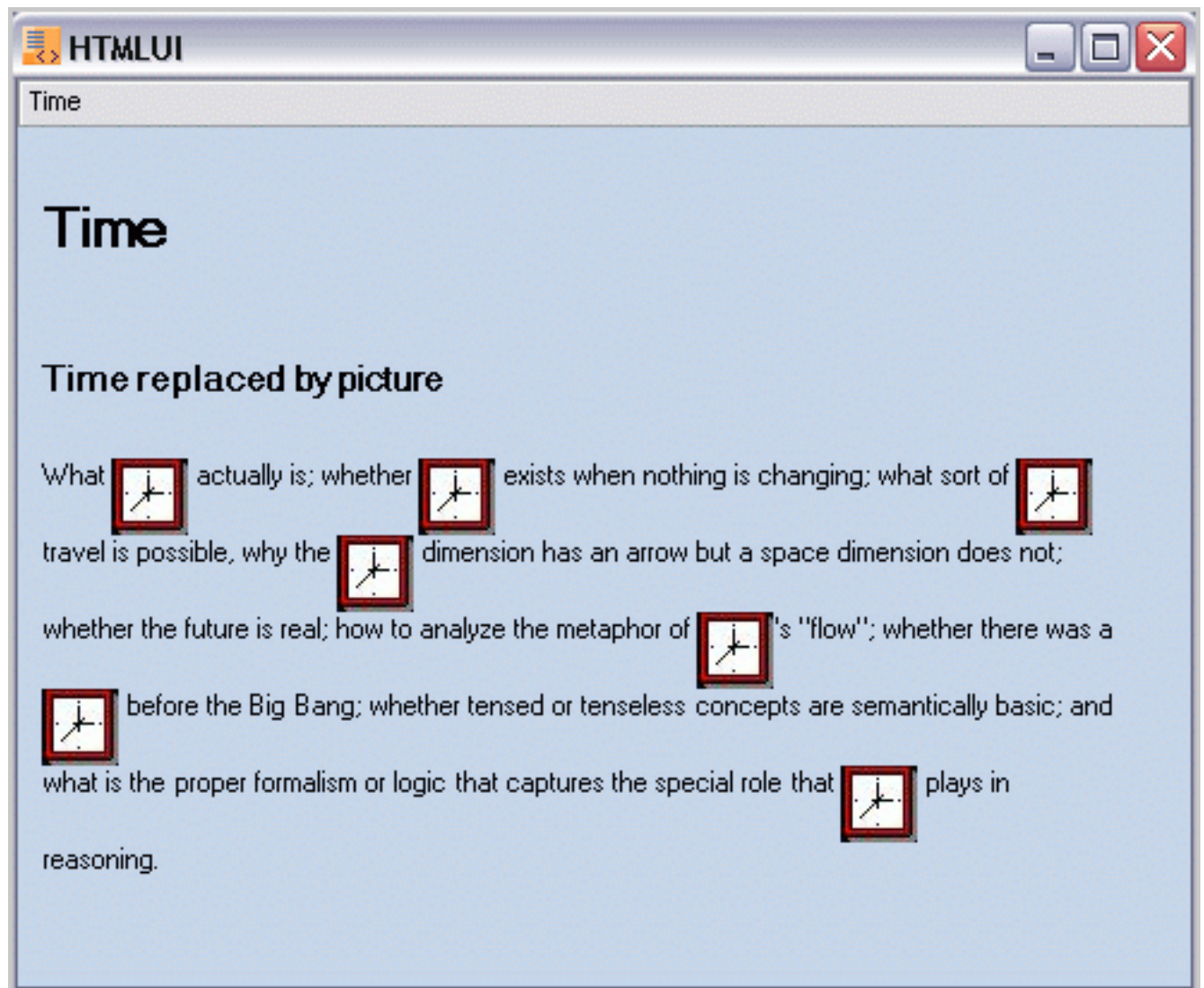


Figure 45: Formatting HTML Elements by using HTMLUI Control

4.16.1 ElementFormat Sample

This sample illustrates Element Formatting in HTMLUI.

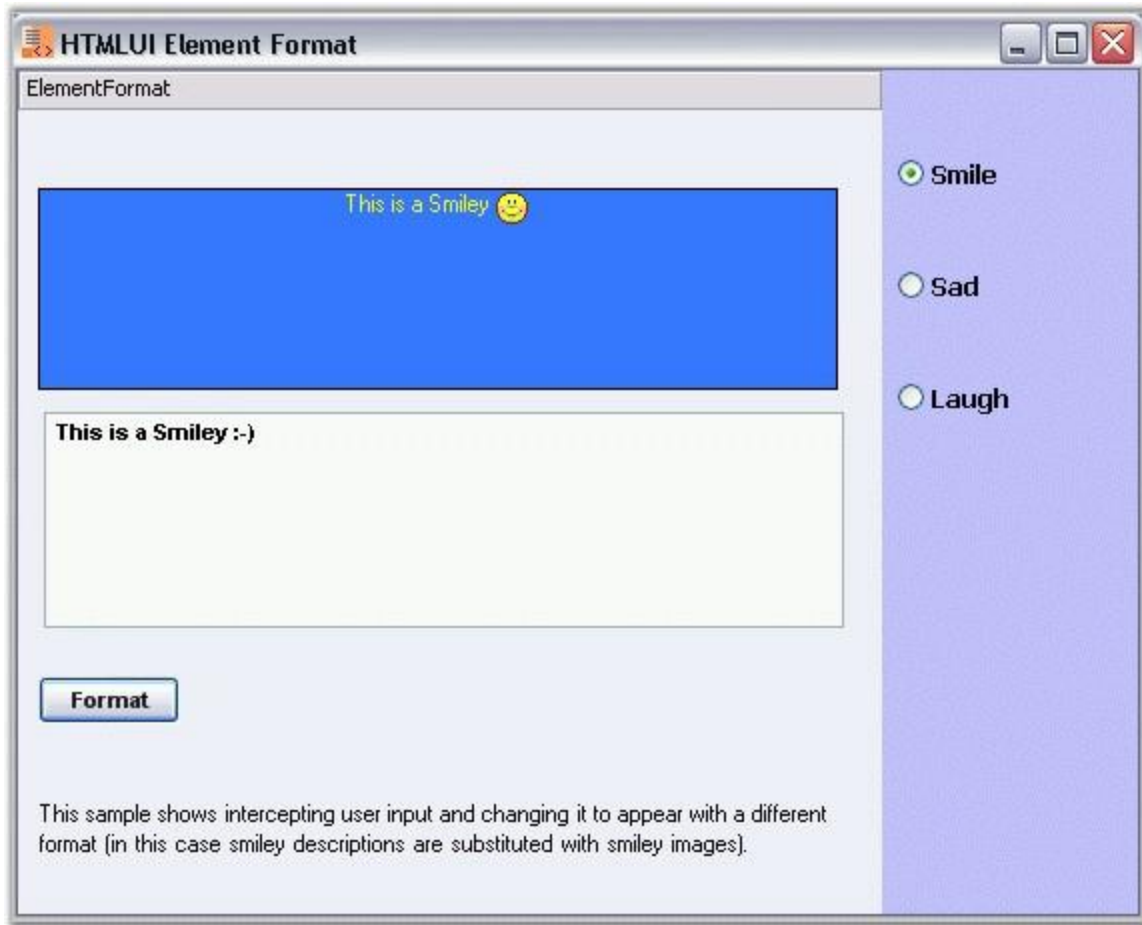


Figure 46: ElementFormat Sample

4.17 Exporting

Essential HTMLUI supports the export of HTML documents. These documents that are available in the HTMLUI control can be exported as images. The HTMLUI control uses the **InputHTML** class to render the HTML document and then converts the available document to Bitmaps. The following code snippet illustrates this.

```
[C#]

private void button1_Click(object sender, System.EventArgs e)
{
    Bitmap bmp = CreateBitmap();
    bmp.Save(@"C:\Myprojects\Bitmap.bmp");
    bmp.Dispose();
}
```

```

private Bitmap CreateBitmap()
{
    FormatManager manager = new FormatManager( htmluiControl1 );
    InputHTML doc = new InputHTML( @"C:\MyProjects\HTML.htm", manager );
    doc.ClientSize = new Size( 550, 200 );
    htmluiControl1.PrepareDocument( doc );
    return LoadFinished( doc );
}

private Bitmap LoadFinished( InputHTML document )
{
    Bitmap bmp = new Bitmap( 450, 500 );
    Graphics g = Graphics.FromImage( bmp );
    PaintEventArgs args = new PaintEventArgs( g, new Rectangle( 0, 0, 450, 500 ) );
    Point startPoint = new Point( -document.Margins.Right, -document.Margins.Top );
    Size oldSize = document.ClientSize;
    document.ClientSize = document.RenderRoot.Size;
    document.Draw( args, startPoint );
    g.Dispose();
    document.ClientSize = oldSize;
    return bmp;
}

```

[VB.NET]

```

Private Sub button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Dim bmp As Bitmap = CreateBitmap()
    bmp.Save("C:\Myprojects\Bitmap.bmp")
    bmp.Dispose()
End Sub

Private Function CreateBitmap() As Bitmap
    Dim manager As FormatManager = New FormatManager(htmluiControl1)
    Dim doc As InputHTML = New InputHTML("C:\MyProjects\HTML.htm", manager)
    doc.ClientSize = New Size(550, 200)
    htmluiControl1.PrepareDocument(doc)
    Return LoadFinished(doc)
End Function

Private Function LoadFinished(ByVal document As InputHTML) As Bitmap
    Dim bmp As Bitmap = New Bitmap(450, 500)
    Dim g As Graphics = Graphics.FromImage(bmp)
    Dim args As PaintEventArgs = New PaintEventArgs(g, New Rectangle(0, 0, 450, 500))
    Dim startPoint As Point = New Point(-document.Margins.Right, -
    document.Margins.Top)
    Dim oldSize As Size = document.ClientSize
    document.ClientSize = document.RenderRoot.Size
    document.Draw(args, startPoint)
    g.Dispose()

```

```

document.ClientSize = oldSize
Return bmp
End Function

```

4.17.1 HTMLUIExporting Sample

This sample illustrates the export of HTML documents loaded into the HTMLUI to images.

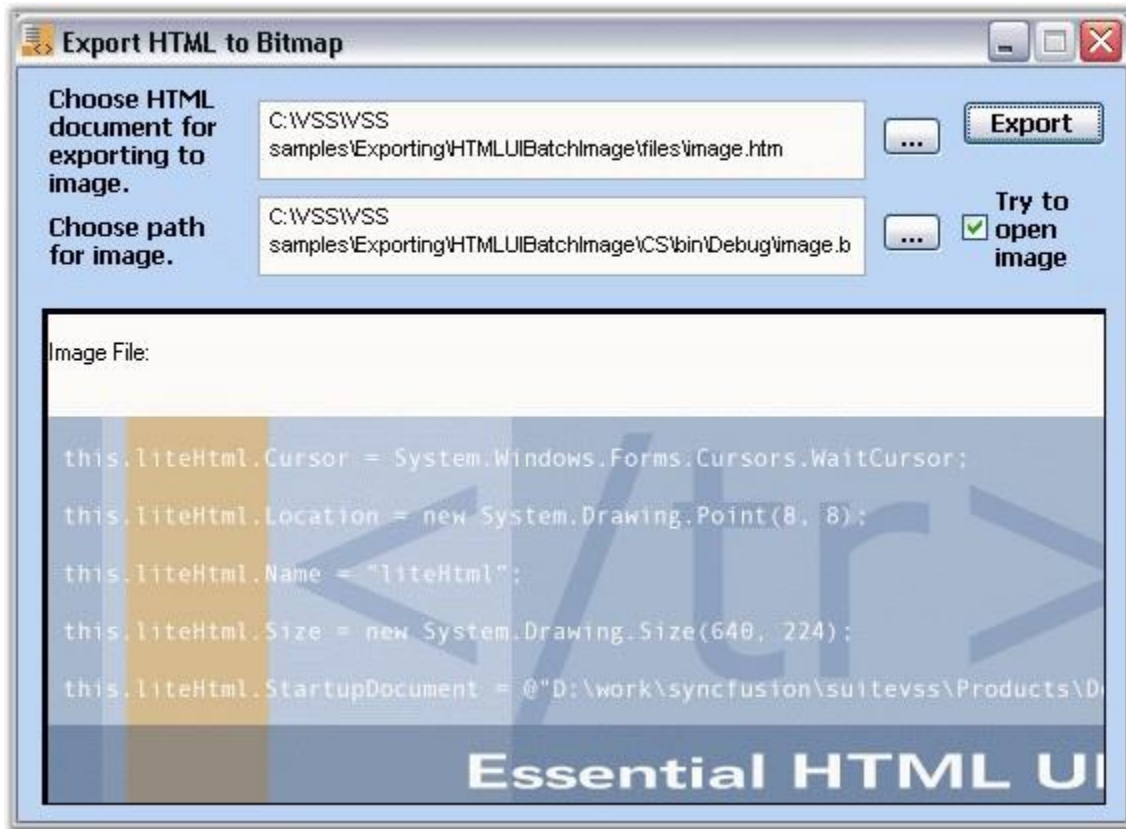


Figure 47: HTMLUIExporting Sample

4.18 Keyboard

The HTMLUI control also supports usage of keyboards for navigating through the links inside a HTML document. Like in popular browsers, HTMLUI control uses the **TAB** key for shifting the focus on the links.

4.18.1 HTMLUIKeyboard Sample

This sample shows how elements in the document can be focused by using the Keyboard support in HTMLUI.

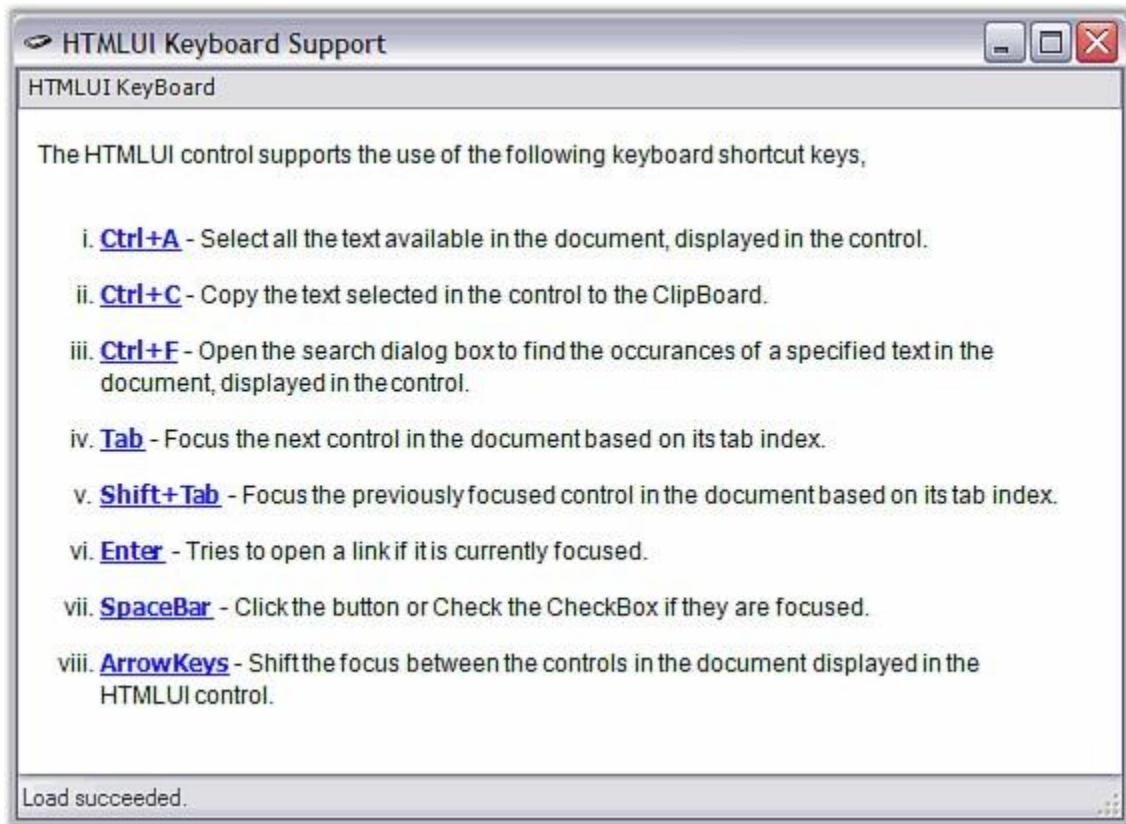


Figure 48: HTMLUIKeyboard Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\ Samples\2.0\HTMLUI UserInteraction\HTMLUIKeyboard

4.19 Printing

The HTMLUI control supports printing that helps the user in developing a hard copy of the document displayed in the HTMLUI control. Also the Print Preview feature lets the user to preview the page before being printed, and change the page according to the requirements.

[C#]

```
//Document initialization
HTMLUIPrintDocument pd = new HTMLUIPrintDocument(this.htmluiControll1.Document);

//Print Preview
PrintPreviewDialog dlg = new PrintPreviewDialog();
dlg.Document = pd;
dlg.ShowDialog();

//Print
PrintDialog dg = new PrintDialog();
dg.AllowSomePages = true;
dg.Document = pd;
if (dg.ShowDialog() == DialogResult.OK)
pd.Print();
```

[VB.NET]

```
'Document initialization
Dim pd As New HTMLUIPrintDocument(Me.HtmluiControll1.Document)

'Print Preview
Dim dlg As New PrintPreviewDialog()
dlg.Document = pd
dlg.ShowDialog()

'Print
Dim dg As New PrintDialog()
dg.AllowSomePages = True
dg.Document = pd
If dg.ShowDialog() = DialogResult.OK Then
pd.Print()
End If
```

Along with printing feature, HTMLUI control supports previewing of the document before printing. This following code snippet shows how the print preview feature is enabled in HTMLUI.

[C#]

```
// Document Initialization
private void PrintPreViewButton_Click(object sender, System.EventArgs e)
{
    HTMLUIPrintDocument pd = new HTMLUIPrintDocument(this.htmluiControll1.Document);

    // Initialize the new instance of the System.Window.Forms.PrintPreviewDialog Class
    PrintPreviewDialog dlg = new PrintPreviewDialog();
    dlg.Document = pd;
    dlg.ShowDialog();
}
```

```
}
```

[VB.NET]

```
' Document Initialization
Private Sub PrintPreViewButton_Click(ByVal sender As Object, ByVal e As
System.EventArgs)
Dim pd As HTMLUIPrintDocument = New HTMLUIPrintDocument(Me.htmluiControl1.Document)

' Initialize the new instance of the System.Window.Forms.PrintPreviewDialog Class
Dim dlg As PrintPreviewDialog = New PrintPreviewDialog()
dlg.Document = pd
dlg.ShowDialog()
End Sub
```

The following figure shows the Print preview page that appears when the corresponding button is clicked. This illustrates the Printing feature in HTMLUI.

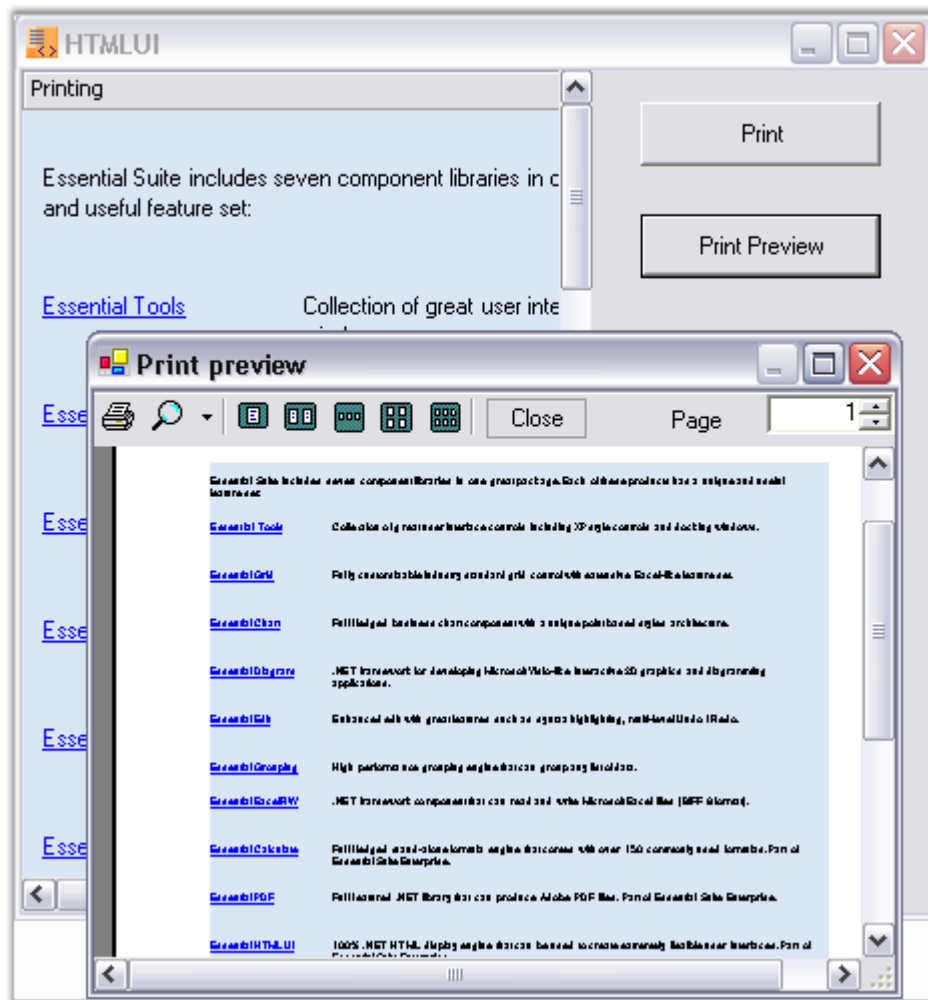


Figure 49: Printing support in the HTMLUI Control

4.19.1 HTMLUIPrinting Sample

This sample shows how an HTML document available in the HTMLUI can be printed.

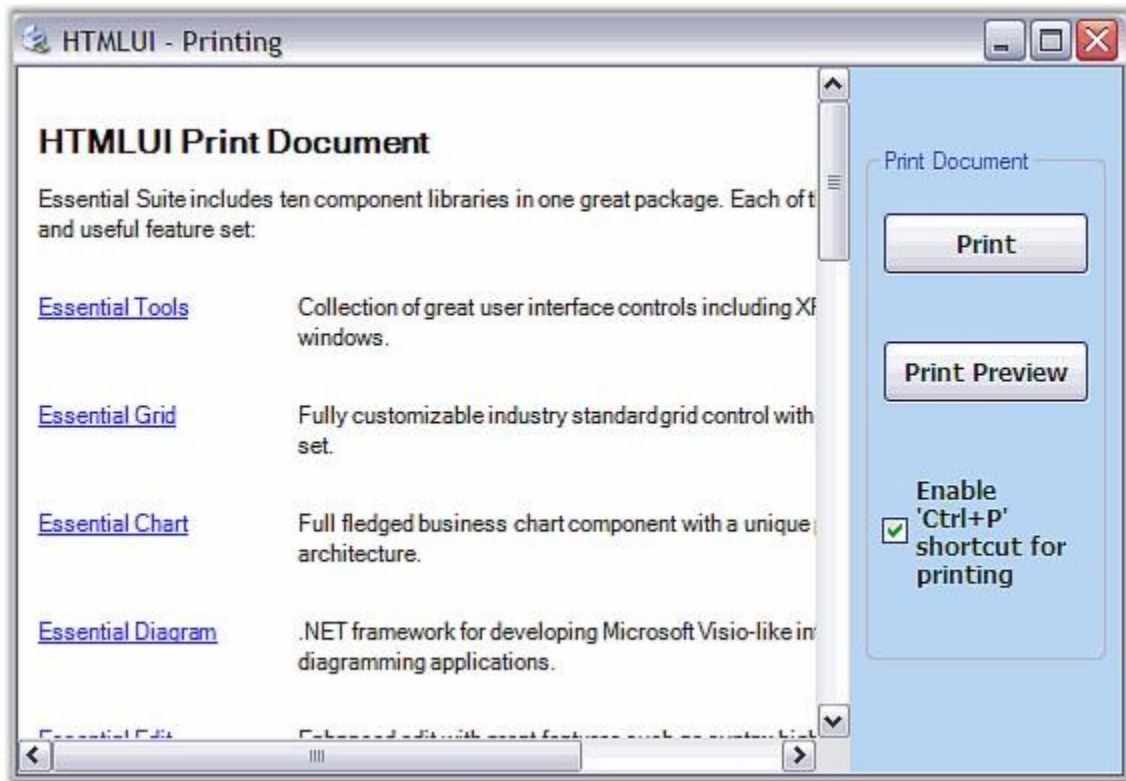


Figure 50: HTMLUIPrinting Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\ Samples\2.0\HTMLUI Exporting\HTMLUIPrinting

4.20 Searching

Like in popular browsers, the HTMLUI control helps the users to search for a given text in the document displayed in the HTMLUI control. The HTMLUI control uses the **DisplayFindForm** method for this purpose. This feature comes with an **Updown** search and also the **Matchcase** search that helps the user to easily find the required text from the displayed document.

The **CTRL+F** shortcut can also be used for enabling this feature.

[C#]

```
// Display the Find form for searching the text content of the HTMLUI control's
current document
this.htmluiControl1.DisplayFindForm();
```

[VB.NET]

```
// Display the Find form for searching the text content of the HTMLUI control's
current document
Me.htmluiControl1.DisplayFindForm()
```

4.20.1 HTMLUISearching Sample

This sample shows how a text can be searched in a document loaded into the HTMLUI.

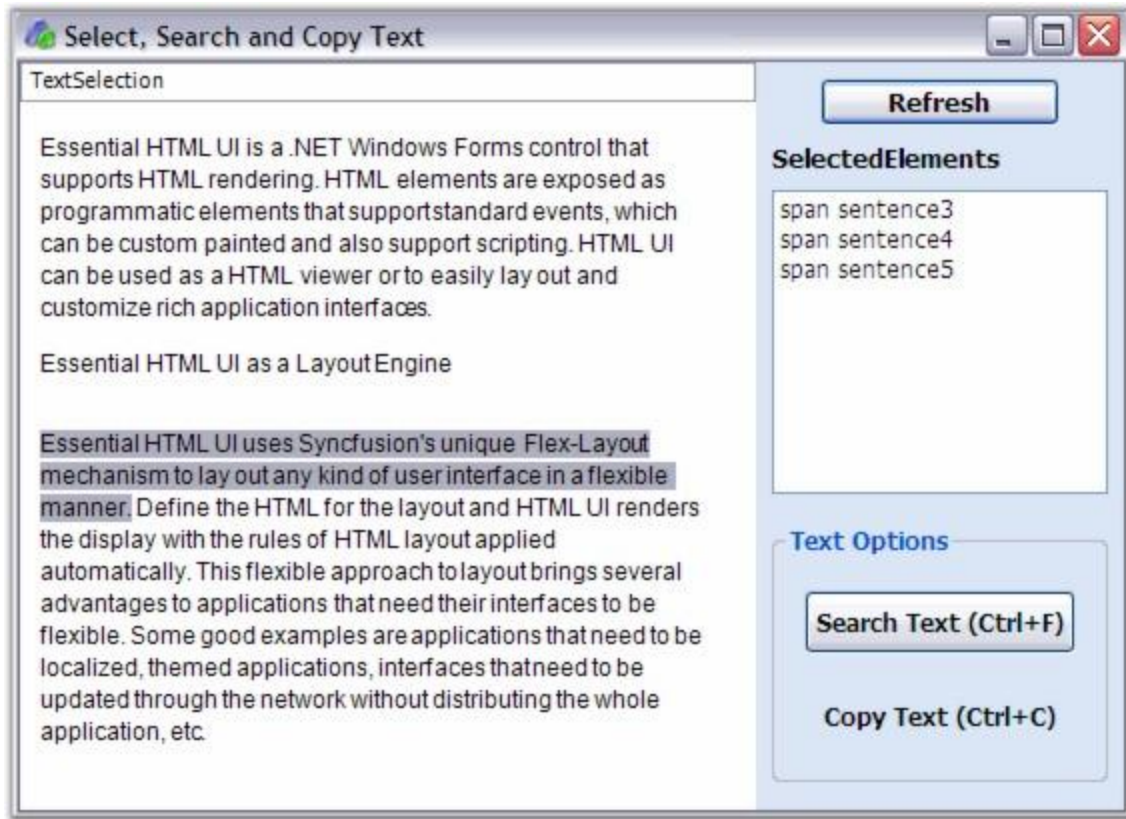


Figure 51: HTMLUISearching Sample

4.21 Scrolling

The **Scroll** property of the HTMLUI control helps in loading large HTML documents. This scroll property can be set as per the application. Also the HTMLUI control allows programmatic scrolling to a particular location or element with its extensive scroll properties support.

[C#]

```
// Scroll controls in such way that the specified element is visible
IHTMLElement elem = this.htmluiControl1.Document.GetElementById("pre");
this.htmluiControl1.ScrollToElement(elem);
```

[VB.NET]

```
' Scroll controls in such way that the specified element is visible
Private elem As IHTMLElement = Me.htmluiControl1.Document.GetElementById("pre")
Me.htmluiControl1.ScrollToElement(elem)
```

4.21.1 HTMLUIAutoScroll Sample

This sample illustrates the Scrolling feature supported in HTMLUI.

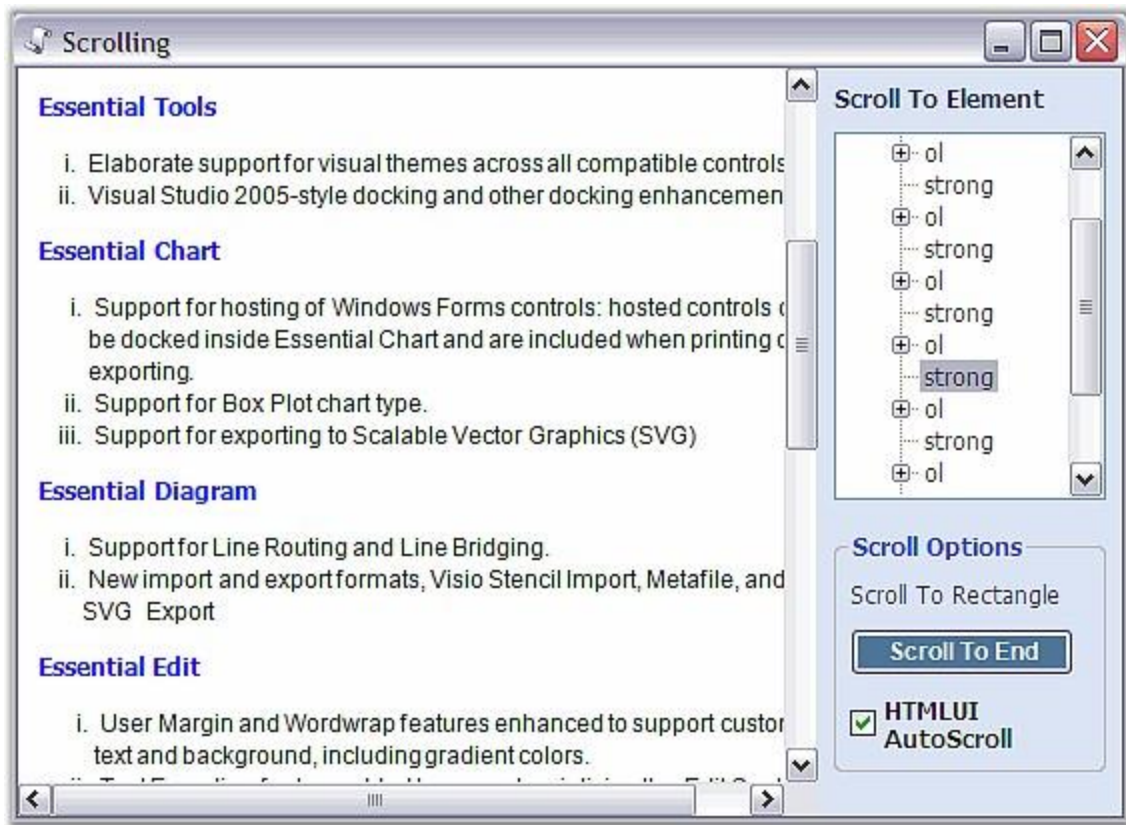


Figure 52: HTMLUIScrolling Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0\HTMLUI UserInteraction\HTMLUIScrolling

4.22 Scripting

Essential HTMLUI has extensive support to Scripts. Scripting involves creating self-contained documents. The concept behind this is to make the interface easier just by loading different HTML documents that contain the logic in themselves.

[HTML]

```

<html>
  <body bgColor="#edf0f7">
    <p>
      <input type="text" id="txt"></input>
    </p>
    <script language="C#">
      using System;
      using System.IO;
      using System.Xml;
      using System.Windows.Forms;
      using System.Drawing;
      using System.Collections;
      using Syncfusion.Windows.Forms.HTMLUI;

      namespace Syncfusion
      {
        public class Script
        {
          INPUTElementImpl scripttext;
          Hashtable htmlelements = new Hashtable();

          /* Initializes script.*/
          public static Script OnScriptStart()
          {
            return new Script();
          }

          public Script()
          {
            htmlelements = Global.Document.GetElementsByUserIdHash();
            scripttext = htmlelements["txt"] as INPUTElementImpl;

            // User control property sets the user control instance for the particular input
            // element declared by the user
            scripttext.UserControl.CustomControl.Text = "This is a sample for scripting";
          }
        }
      }
    </script><br/>
  </body>
</html>

```

4.22.1 HTMLUIScripting Sample

This sample illustrates the support of self-contained HTML documents in HTMLUI.

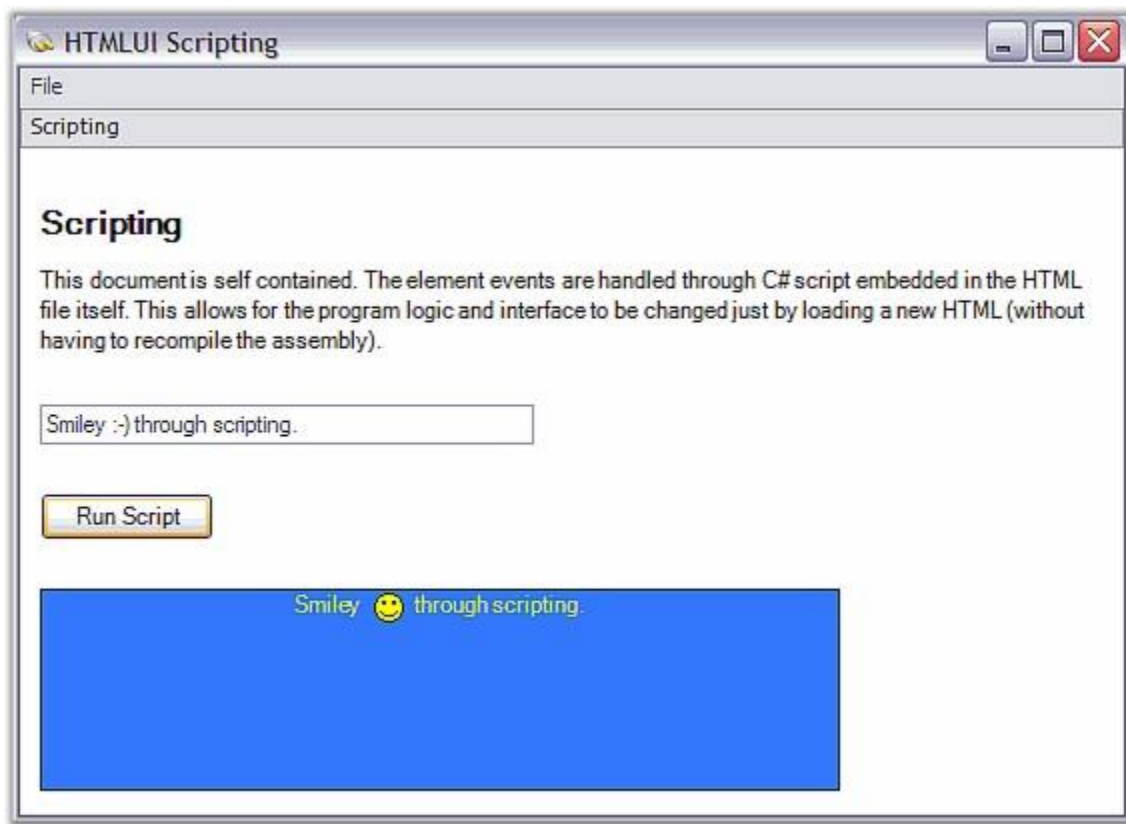


Figure 53: HTMLUIScripting Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\Syncfusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0\HTML Renderer\HTMLUIScripting

4.23 Text Selection

An interesting feature of the HTMLUI control is its ability to access the selected text. This feature helps the user to select required texts available in the HTMLUI control and use the selected text in the applications. The **SelectedText** property of the HTMLUI control is used for this purpose.

[C#]

```
// Return the selected text displayed in the control
this.label1.Text = this.htmluiControl1.SelectedText;
```

[VB.NET]

```
' Return the selected text displayed in the control  
Me.label1.Text = Me.HtmluiControl1.SelectedText
```

CopyTextToClipboard

The HTMLUI control allows the user to copy the text selected in the HTMLUI control to the Clipboard, and paste it in other applications. The following code snippet shows how this feature is implemented with the HTMLUI control.

[C#]

```
string text = this.htmluiControl.SelectedText.ToString();  
if (text != "")  
{  
    //Copying the selected text to the Clipboard  
    Clipboard.SetDataObject(text);  
}
```

[VB.NET]

```
Private text As String = Me.htmluiControl.SelectedText.ToString()  
If text <> "" Then  
  
    ' Copying the selected text to the Clipboard  
    Clipboard.SetDataObject(text)  
End If
```

4.23.1 HTMLUITextSelection Sample

This sample demonstrates the support for Text Selection in HTMLUI.

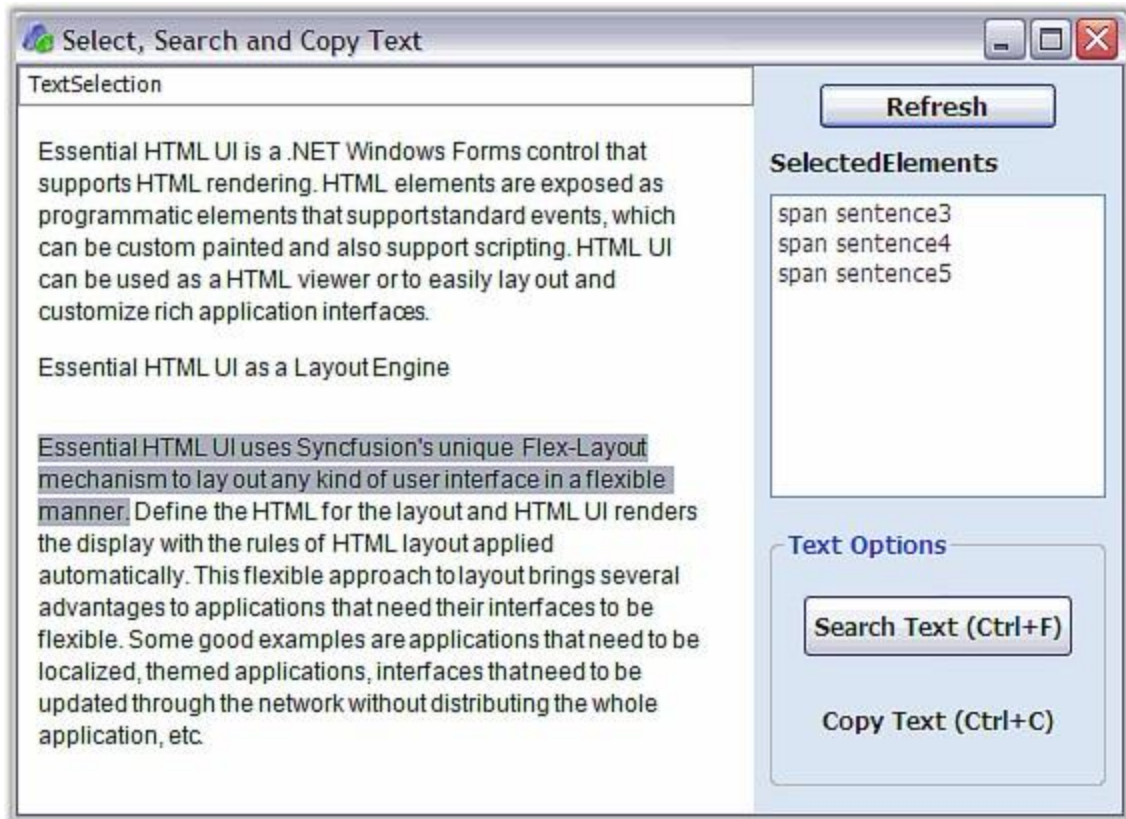


Figure 54: HTMLUITextSelection Sample

By default, this sample can be found under the following location:

C:\Documents and Settings\username\My Documents\SynCFusion\EssentialStudio\Version Number\Windows\HTMLUI.Windows\Samples\2.0\HTMLUI UserInteraction\HTMLUITextSelection

5 Frequently Asked Questions

This section discusses various frequently asked questions with their answers, examples and code snippets.

5.1 How To Access All the Child Elements Of an HTML Element In the HTMLUI Control?

The **IHTMLElement.Children** property of any IHTMLElement, collects all the child elements of a specified HTML element inside an **IHTMLElementsCollection**. You can access the elements needed for your conditions from this collection.

The following code snippet illustrates how the child elements of the Body element in the given HTML document are searched to access elements containing the OnClick attribute and how a Click event is attached to those elements.

[HTML]

```
<html>
  <head>
    <style>.nav{background-color:#dae5f5}</style>
  </head>
  <body>
    <p/>
    
    <p/>
    <div>This is a sample</div>
  </body>
</html>
```

[C#]

```
private void htmluiControl1_LoadFinished(object sender, System.EventArgs e)
{
    //Getting the body element in the HTML document
    IHTMLElement[] body = this.htmluiControl1.Document.GetElementsByName("body");
    //Collecting the children of the body element in a collection
    IHTMLElementsCollection elem = body[0].Children;
    foreach (IHTMLElement child in elem)
    {
        //searching for the children containing the OnClick attribute
        if (child.Attributes.Contains("ONCLICK") == true)
        {
            //Click event declaration for current children
        }
    }
}
```



```
child.Click += new EventHandler(child_Click);
}
}
}

private void child_Click(object sender, EventArgs e)
{
    BubblingEventArgs bargs = HTMLUIControl.GetBublingEventArgs(e);
    //Accessing the element that is sending the event
    BaseElement elem = bargs.RootSender as BaseElement;
    //Validating the element for execution
    if (elem.ID == "img1" && elem is IMGElementImpl)
    {
        if (elem.Attributes["src"].Value == "sync.jpg")
            elem.Attributes["src"].Value = "syncfusion.gif";
        else
            elem.Attributes["src"].Value = "sync.jpg";
        this.htmluiControll1.ScrollToElement(elem);
    }
}
```

[VB.NET]

```
Private Sub htmluiControll1_LoadFinished(ByVal sender As Object, ByVal e As
System.EventArgs)
    'Getting the body element in the HTML document
    Dim body As IHTMLElement() =
        Me.htmluiControll1.Document.GetElementsByName("body")

    'Collecting the children of the body element in a collection
    Dim elem As IHTMLElementCollection = body(0).Children
    For Each child As IHTMLElement In elem

        'searching for the children containing the OnClick attribute
        If child.Attributes.Contains("ONCLICK") = True Then

            'Click event declaration for current children
            AddHandler child.Click, AddressOf child_Click
        End If
    Next child
End Sub

Private Sub child_Click(ByVal sender As Object, ByVal e As EventArgs)
    Dim bargs As BubblingEventArgs = HTMLUIControl.GetBublingEventArgs(e)
    'Accessing the element that is sending the event
    Dim elem As BaseElement = CType(IIf(EOF(bargs.RootSender Is BaseElement,
        bargs.RootSender, Nothing), BaseElement)

    'Validating the element for execution
    If elem.ID = "img1" AndAlso EOF(elem Is IMGElementImpl Then
```



```
If elem.Attributes("src").Value = "sync.jpg" Then
    elem.Attributes("src").Value = "syncfusion.gif"
Else
    elem.Attributes("src").Value = "sync.jpg"
End If
Me.htmluiControl1.ScrollToElement(elem)
End If
End Sub
```

5.2 How To Access the HTML Elements Loaded Into the Control?

The HTML elements are accessed in HTMLUI for developing advanced UIs. The HTML elements are collected in a collection class. When the **Hashtable** is used as a collection class, it stores the HTML elements with a key, specific for each element.

These elements are then assigned to the code variables. The **Code Variables** are the objects of the classes that are responsible for the functionality of the tag elements. These classes contain the definitions for the properties, methods and events for the tag elements. These variables will be used in accessing the HTML elements inside the code.

The following HTML document shows an input tag textbox element with an id as the key for accessing it in the Hashtable.

```
[HTML]

<html>
  <body>
    <input type="text" id="txt"/>
  </body>
</html>
```

The following code snippet illustrates accessing the HTML elements from the above document.

[C#]

```
//declaring the code variable of the tag element
//INPUTElementImpl is responsible for the INPUT tag in HTMLUI
INPUTElementImpl text;

//Collection class accessing the HTML document with the userID as the key
Hashtable htmelements =
this.htmluiControll1.Document.GetElementsByUserIdHash();

//Code variable assigned to the html element with the help of the element's id
as //the key
this.text = htmelements["txt"] as INPUTElementImpl;

//usage of the code variable inside the project
this.text.UserControl.CustomControl.Text = "HTML Elements in HTMLUI";
```

[VB.NET]

```
'declaring the code variable of the tag element
'INPUTElementImpl is responsible for the INPUT tag in HTMLUI
Private text As INPUTElementImpl

'Collection class accessing the HTML document with the userID as the key
Private htmelements As Hashtable =
Me.HtmluiControll1.Document.GetElementsByUserIdHash()

'Code variable assigned to the html element with the help of the element's id
as 'the key
Me.Text = CType(IIf(OfType htmelements("txt") Is INPUTElementImpl,
htmelements("txt"), Nothing), INPUTElementImpl)

'usage of the code variable inside the project
Me.Text.UserControl.CustomControl.Text = "HTML Elements in HTMLUI"
```

5.3 How To Access the Inner HTML Text Of the Current HTML Element In the HTMLUI Control?

You can access the inner HTML text of the current HTML element in the HTMLUI control by using the **InnerHTML** property of the HTMLUI control. This property also allows access to the child elements of the HTML elements.

The following HTML document contains a div element. The code snippet shows how the inner text of the element is accessed and displayed in the output at run time.

[HTML]

```
<!-- HTML Document -->
<html>
    <body>
        <div id="div1">
            Have an issue you need to contact Syncfusion about? Use our
            state of the art incident management
            system - Direct-Trac.
        </div>
    </body>
</html>
```

[C#]

```
Hashtable htmelements =
this.htmluiControll.Document.GetElementsByIdHash();
DIVElementImpl div1 = this.htmelements["div1"] as DIVElementImpl;
MessageBox.Show(div1.InnerHTML.ToString());
```

[VB.NET]

```
Private htmelements As Hashtable =
Me.htmluiControll.Document.GetElementsByIdHash()
Private div1 As DIVElementImpl = CType(If(OfType Me.htmelements("div1") Is
DIVElementImpl,
Me.htmelements("div1"), Nothing), DIVElementImpl)
MessageBox.Show(div1.InnerHTML.ToString())
```

5.4 How To Access the Name Of an HTML Element At Run-time?

The **element.Name** property gets the name of the tag that defines the element as an attribute, and not the name of the element defined by the user. You can access the name of the element with the help of the **element.Attributes** property.

The following HTML document illustrates how an input element with a name is declared and accessed in HTMLUI.

[HTML]

```
<html>
    <body>
        <p>
            <input type="text" id="txt1" name="textboxOne"
size="20"></input>
```

```
        </p>
    </body>
</html>
```

[C#]

```
INPUTElementImpl textBox1;
Hashtable htmelements =
this.htmluiControll1.Document.GetElementsByUserIdHash();

//accessing the input element
textBox1 = htmelements["txt1"] as INPUTElementImpl;

//accessing the tag name of the element
Console.WriteLine("Tag Name of TextBox1:\n"+ textBox1.Name.ToString());

//accessing the attribute name of the element
Console.WriteLine("Name given for TextBox1:\n"+
textBox1.Attributes["name"].Value);
```

[VB.NET]

```
Private textBox1 As INPUTElementImpl
Private htmelements As Hashtable =
Me.htmluiControll1.Document.GetElementsByUserIdHash()

'accessing the input element
Private textBox1 = CType(If(TypeOf htmelements("txt1") Is INPUTElementImpl,
htmelements("txt1"), Nothing), INPUTElementImpl)

'accessing the tag name of the element
Console.WriteLine("Tag Name of TextBox1:" & Constants.vbLf+
textBox1.Name.ToString())

'accessing the attribute name of the element
Console.WriteLine("Name given for TextBox1:" & Constants.vbLf+
textBox1.Attributes("name").Value)
```

5.5 How To Add an Attribute To an HTML Element And Change Its Value At Run-time?

You can add an attribute to an HTML element using the **Add** method of the **Attributes** property, as shown in the following code snippet.

[C#]

```
// textBox is an INPUT element of type 'text' in the HTML document
if(this.textBox.Attributes.Contains("style") == false)
this.textBox.Attributes.Add("style");
```

[VB.NET]

```
' textBox is an INPUT element of type 'text' in the HTML document
If Me.textBox.Attributes.Contains("style") = False Then
Me.textBox.Attributes.Add("style")
End If
```

You can change the value of an element's attribute at run time by using the **Value** property of that particular attribute, as shown in the below code snippet.

[C#]

```
// textBox is an INPUT element of type 'text' in the HTML document
this.textBox.Attributes["style"].Value = "background-color:red";
```

[VB.NET]

```
' textBox is an INPUT element of type 'text' in the HTML document
Private Me.textBox.Attributes("style").Value = "background-color:red"
```

5.6 How To Change a Characteristic Of an HTML Element Before It Is Displayed?

The characteristic of an element can be easily changed in the **PreRenderDocument** event of the HTMLUI control.

The PreRenderDocument event is raised when the elements in the HTML document are created in the HTMLUI control, but their size and location are not calculated yet.

[HTML]

```
<html>
  <body bgcolor="#c4d6e9">
    <h1>Syncfusion</h1>
    <h4>.NET Essentials</h4>
    <p>
      
    </p>
    <p>
```

Essential Studio includes ten component libraries in one great package. Each of these products has a unique and useful feature set. Syncfusion aims to provide customers with the utmost satisfaction and value in using Syncfusion and Microsoft technologies through our technical consulting and training services.

```
</p>
</body>
</html>
```

The following snippet shows how an image reference is changed for a page in the HTMLUI at run time, by using the `PrerenderDocument` event.

[C#]

```
Hashtable htmelements = new Hashtable();

// Event Declaration.
this.htmluiControll.PreRenderDocument +=
newSyncfusion.Windows.Forms.HTMLUI.PreRenderDocumentEventHandler(this.htmluiCon
troll.PreRenderDocument);

// Event that is to be raised when a tree of element has been created and their
size and location have
// not been calculated yet.
private void htmluiControll.PreRenderDocument(object
sender, PreRenderDocumentArgs e)
{
    // Create and Return the Hash Table where key is Tag name(Element.Name) and
Value is array of
// elements with Current name.
this.htmelements = e.Document.GetElementsByNameHash();
ArrayList imgs = this.htmelements["img"] as ArrayList;
foreach(BaseElement elem in imgs)
{
    string oldValue = elem.Attributes["src"].Value;
    string newValue = @"C:\MyProjects\ImageHandling\newImage.jpg";

    // Replace or change the value of the attribute in the element at run time.
    elem.Attributes["src"].Value = oldValue.Replace(elem.Attributes["src"].Value,
newValue);
    Console.WriteLine(elem.Attributes["src"].Value);
}
}
```

[VB.NET]

```
Private htmelements As Hashtable = New Hashtable()

' Event Declaration.
Me.htmluiControll.PreRenderDocument += New
```

```
Syncfusion.Windows.Forms.HTMLUI.PreRenderDocumentEventHandler(Me.htmluiControll1
_PreRenderDocument)

' Event that is to be raised when a tree of element has been created and their
size and location have
' not been calculated yet.
Private Sub htmluiControll1_PreRenderDocument(ByVal sender As Object, ByVal e As
PreRenderDocumentArgs)

' Create and Return the Hash Table where key is Tag name(Element.Name) and
Value is array of
' elements with Current name.
Me.htmlelements = e.Document.GetElementsByNameHash()
Dim imgs As ArrayList = CType(If(OfType Me.htmlelements("img") Is ArrayList,
Me.htmlelements("img"), Nothing), ArrayList)
For Each elem As BaseElement In imgs
Dim oldValue As String = elem.Attributes("src").Value
Dim newValue As String = @"C:\MyProjects\ImageHandling\newImage.jpg";

' Replace or change the value of the attribute in the element at run time.
elem.Attributes("src").Value = oldValue.Replace(elem.Attributes("src").Value,
newValue)
Console.WriteLine(elem.Attributes("src").Value)
Next elem
End Sub
```

The following figure illustrates this behavior where the oldImage has been replaced by newImage.

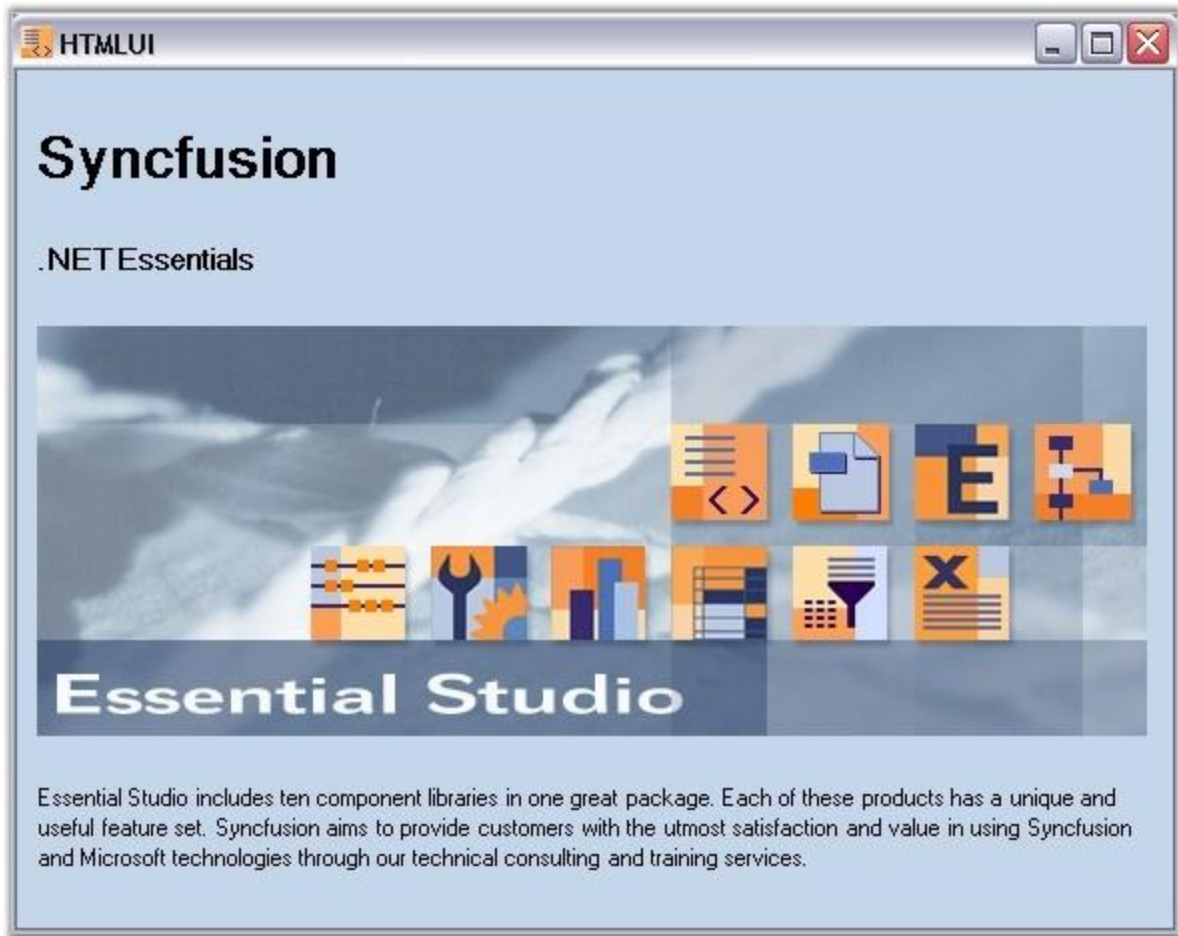


Figure 55: Changing Image Reference by using the PreRenderDocument Event of the HTMLUI Control

5.7 How To Change the Default Font Used For Rendering the HTML Document In the HTMLUI Control?

HTMLUI uses a default font to render the text from the HTML document, in cases where there are no specifications for the font to be used. You can change this default font by using the **DefaultFormat.Font** property, written while initializing the HTMLUI control.

[C#]

```
htmluiControl1 = new Syncfusion.Windows.Forms.HTMLUI.HTMLUIControl();  
htmluiControl1.DefaultFormat.Font = new Font("Pristina", 16);
```


[VB.NET]

```
Private htmluiControl1 = New Syncfusion.Windows.Forms.HTMLUI.HTMLUIControl()  
Private htmluiControl1.DefaultFormat.Font = New Font("Pristina",16)
```

5.8 How To Delete an HTML Element From a Document Loaded Into the HTMLUI Control At Run-time?

The HTML elements loaded in the HTMLUI control are collected in the **IHTMLElementsCollection**. You can make use of the **Remove** method of the **IHTMLElementsCollection** Interface to remove an element from the current collection, and the **Refresh** method to redraw the HTMLUI control with changes updated in the current document.

The following HTML document contains a textbox and a button element. The following code snippet shows how the textbox and the button are removed from the HTMLUI control's display at run time.

[HTML]

```
<!-- HTML document -->  
<html>  
    <body>  
        <p>  
            <input type="text" id="txt1"></input>  
        </p>  
        <p>  
            <input type="button" id="btn1" value="Button1"></input>  
        </p>  
    </body>  
</html>
```

[C#]

```
IHTMLElement txt1;  
IHTMLElement btn1;  
  
private System.Windows.Forms.Button button1;  
button1.Click += new System.EventHandler(this.button1_Click);  
htmluiControl1.LoadFinished += new  
System.EventHandler(this.htmluiControl1_LoadFinished);  
  
private void htmluiControl1_LoadFinished(object sender, System.EventArgs e)  
{  
    this.txt1 = this.htmluiControl1.Document.GetElementByUserId("txt1");  
    this.btn1 = this.htmluiControl1.Document.GetElementByUserId("btn1");  
}
```

```
private void button1_Click(object sender, System.EventArgs e)
{
    this.text1.Parent.Children.Remove(this.text1);
    this.btn1.Parent.Children.Remove(this.btn1);
    this.htmluiControl1.Refresh();
}
```

[VB.NET]

```
Private text1 As IHTMLElement
Private btn1 As IHTMLElement

Private button1 As System.Windows.Forms.Button
Private button1.Click += New System.EventHandler(Me.button1_Click)
Private htmluiControl1.LoadFinished += New
System.EventHandler(Me.htmluiControl1_LoadFinished)

Private Sub htmluiControl1_LoadFinished(ByVal sender As Object, ByVal e As
System.EventArgs)
    Me.text1 = Me.htmluiControl1.Document.GetElementById("txt1")
    Me.btn1 = Me.htmluiControl1.Document.GetElementById("btn1")
End Sub

Private Sub button1_Click(ByVal sender As Object, ByVal e As System.EventArgs)
    Me.text1.Parent.Children.Remove(Me.text1)
    Me.btn1.Parent.Children.Remove(Me.btn1)
    Me.htmluiControl1.Refresh()
End Sub
```

5.9 How To Enable the HTMLUI Control To Load HTML Documents That Have Been Dragged Onto the Control?

In order to enable the HTMLUI control to load HTML documents that have been dragged onto the control, you have to set the **AllowDrop** property of the HTMLUI control to **True**. This property helps in supporting the drag-and-drop operation in the HTMLUI control.

During the drag-and-drop operation, the file name of the document along with the location is collected, and when the mouse button is released, the specified document is loaded from the mentioned location through the **LoadHTML** method of the HTMLUI control.

[C#]

```
//To support drag events to the control
this.htmluiControl1.AllowDrop = true;

//DragDrop and DragEnter events declaration
this.htmluiControl1.DragDrop += new
System.Windows.Forms.DragEventHandler(this.htmluiControl1_DragDrop);
this.htmluiControl1.DragEnter += new
System.Windows.Forms.DragEventHandler(this.htmluiControl1_DragEnter);

private void htmluiControl1_DragEnter(object sender,
System.Windows.Forms.DragEventArgs e)
{
    if(e.Data.GetDataPresent (DataFormats.FileDrop))
        //Specifying the drop effect
        e.Effect = DragDropEffects.All;
    else
        e.Effect = DragDropEffects.None;
}

private void htmluiControl1_DragDrop(object sender,
System.Windows.Forms.DragEventArgs e)
{
    //Specifying the drop format and collecting the file name
    string[] files = (string[])e.Data.GetData("FileDrop", false);
    foreach (string fileName in files)
    {
        //Loading the specified file in to the HTMLUI control
        this.htmluiControl1.LoadHTML(fileName);
    }
}
```

[VB.NET]

```
'To support drag events to the control
Private Me.htmluiControl1.AllowDrop = True

'DragDrop and DragEnter events declaration
Private Me.htmluiControl1.DragDrop += New
System.Windows.Forms.DragEventHandler(Me.htmluiControl1_DragDrop)
Private Me.htmluiControl1.DragEnter += New
System.Windows.Forms.DragEventHandler(Me.htmluiControl1_DragEnter)

Private Sub htmluiControl1_DragEnter(ByVal sender As Object, ByVal e As
System.Windows.Forms.DragEventArgs)
    If e.Data.GetDataPresent(DataFormats.FileDrop) Then
        'Specifying the drop effect
        e.Effect = DragDropEffects.All
    Else
        e.Effect = DragDropEffects.None
    End If
End Sub
```

```

        End If
    End Sub

    Private Sub htmluiControl1_DragDrop(ByVal sender As Object, ByVal e As
        System.Windows.Forms.DragEventArgs)
        'Specifying the drop format and collecting the file name
        Dim files As String() = CType(e.Data.GetData("FileDrop", False), String())
        For Each fileName As String In files
            'Loading the specified file in to the HTMLUI control
            Me.htmluiControl1.LoadHTML(fileName)
        Next fileName
    End Sub

```

5.10 How To Enable User Interaction With the HTML Elements

The HTMLUI control supports a rich set of interactivity with the elements displayed. The HTML elements in the HTML document are accessed with the object variables of the respective classes.

HTMLUI control gives a free hand to the user in deciding each and every factor of the element's display. Some include selecting the position of the control, attaching the attributes at run time, attaching events with the element, changing the text inside it at run time, and so on.

The following snippet shows how the elements interact with each other on the execution of their respective events.

[HTML]

```

<html>
  <head>
    <title>HTMLUI Element Interactivity</title>
  </head>
  <body bgcolor="#DBE2F2">
    <p><input type="button" id="btn" value="HTMLButton"/></p>
    <p><div id="div">HTMLUI supports a wide variety of HTML tags that can be
      used to display very rich HTML documents.</div></p>
  </body>
</html>

```

[C#]

```

// Class that is responsible for <input> tag
INPUTElementImpl button;

// Class that is responsible for <div> tag
DIVElementImpl div;

private void htmluiControl1_LoadFinished(object sender, System.EventArgs e)

```

```
{

    Hashtable html = this.htmluiControl1.Document.GetElementsByIdHash();
    this.button = html["btn"] as INPUTElementImpl;
    this.div = html["div"] as DIVElementImpl;
    this.button.Click += new EventHandler(button_Click);

    // Event raised when InnerHtml property changed
    this.div.InnerHTMLChanged += new
    ValueChangedEventArgs(div_InnerHTMLChanged);
}

private void button_Click(object sender, EventArgs e)
{

    // Gets or sets text, the inner part of the element. Inner part includes
    children of the current element.
    this.div.InnerHTML = "HTMLUI is very easy to use with complete designer
    integration. The control is very powerful and rich interfaces can be created
    with minimal coding.";
}

private void div_InnerHTMLChanged(object sender, ValueChangedEventArgs e)
{

    // Returns the HTML text including inner and current element text; also
    current element is added in to
    // the output.
    this.div.Children.Parent.Parent.InnerHTML = this.div.OuterHTML + "<p/><img
    src='HTMLUI.gif'/>";
}
}
```

[VB.NET]

```
' Class that is responsible for <input> tag
Private button As INPUTElementImpl

' Class that is responsible for <div> tag
Private div As DIVElementImpl

Private Sub htmluiControl1_LoadFinished(ByVal sender As Object, ByVal e As
System.EventArgs)

    Dim html As Hashtable = Me.htmluiControl1.Document.GetElementsByIdHash()
    Me.button = CType(If(OfType(html("btn")) Is INPUTElementImpl, html("btn"),
    Nothing), INPUTElementImpl)
    Me.div = CType(If(OfType(html("div")) Is DIVElementImpl, html("div"), Nothing),
    DIVElementImpl)
    AddHandler button.Click, AddressOf button_Click

    // Event raised when InnerHtml property changed
```

```
AddHandler div.InnerHTMLChanged, AddressOf div_InnerHTMLChanged
End Sub

Private Sub button_Click(ByVal sender As Object, ByVal e As EventArgs)

    ' Gets or sets text, the inner part of the element. Inner part includes
    ' children of the current element.
    Me.div.InnerHTML = "HTMLUI is very easy to use with complete designer
    integration. The control is very powerful and rich interfaces can be created
    with " & ControlChars.CrLf & "minimal coding."
End Sub

Private Sub div_InnerHTMLChanged(ByVal sender As Object, ByVal e As
ValueChangedEventArgs)

    ' Returns the HTML text including inner and current element text; also current
    ' element is added in to
    ' the output
    Me.div.Children.Parent.Parent.InnerHTML = Me.div.OuterHTML & "<p/><img
    src='HTMLUI.gif'/>"
End Sub
```

The button and the div elements are obtained. Events are declared for both the elements, and on clicking the button, the text inside the div element changes. When the text inside the div element is changed, an image element is appended to the div element. This shows how interactivity can be handled in HTMLUI.

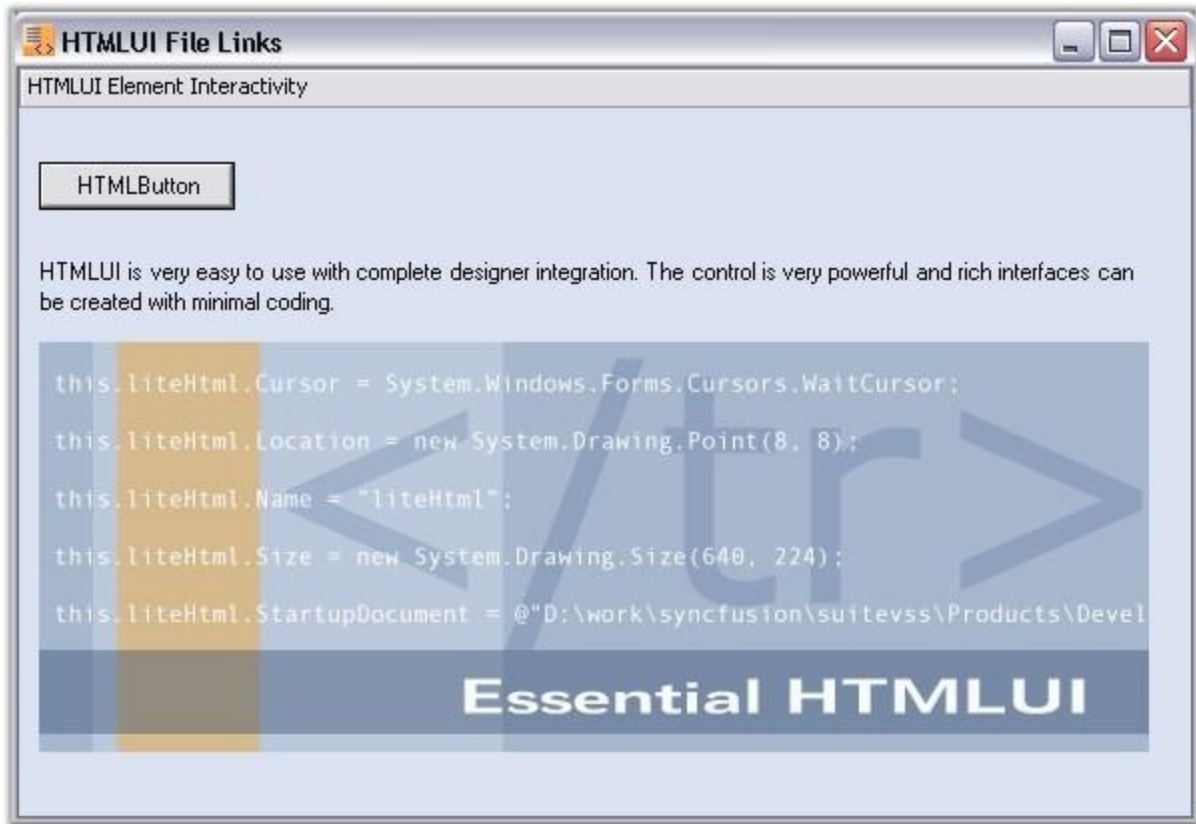


Figure 56: Enabling User Interaction with the HTML Elements in the HTMLUI Control

5.11 How To Get an Object For the Control Present In an HTML Element In the HTMLUI Control?

You can make use of the **GetControlByElement()** method of the **InputHTML** Interface to get an object for the control present in an HTML element in the HTMLUI control. If the HTML element does not contain any control in it, it returns a null value, by default.

[HTML]

```

<html>
  <body>
    <input type="text" id="txt"/>
  </body>
</html>

```

[C#]

```
//Initializing the respective control's object
private System.Windows.Forms.RadioButton htmlRadioButton;

private void htmluiControl1_LoadFinished(object sender, System.EventArgs e)
{
    //Collecting the html elements in a hashtable with their key as id
    Hashtable elements = this.htmluiControl1.Document.GetElementsByUserIdHash();
    BaseElement radioElem = (BaseElement)elements["radio1"];

    //Getting the Control from the html element and assigning it to the required
    object
    //The html element hereafter can be accessed with the help of the
    htmlRadioButton object
    htmlRadioButton = (RadioButton)
    this.htmluiControl1.Document.GetControlByElement(radioElem);
}
```

[VB.NET]

```
'Initializing the respective control's object
Private htmlRadioButton As System.Windows.Forms.RadioButton

Private Sub htmluiControl1_LoadFinished(ByVal sender As Object, ByVal e As
System.EventArgs)
'Collecting the html elements in a hashtable with their key as id
Dim elements As Hashtable =
Me.htmluiControl1.Document.GetElementsByUserIdHash()
Dim radioElem As BaseElement = CType(elements("radio1"), BaseElement)

'Getting the Control from the html element and assigning it to the required
object
'The html element hereafter can be accessed with the help of the
htmlRadioButton object
htmlRadioButton =
CType(Me.htmluiControl1.Document.GetControlByElement(radioElem), RadioButton)
End Sub
```

5.12 How To Load Custom Controls As Part Of the HTML Layout?

An HTML document containing custom controls is shown below.

[HTML]

```
<HTML>
```



```

<HEAD>
<TITLE>HTMLUI CUSTOM CONTROLS</TITLE>
<style>
.tttDisplay { text-decoration: none; color: #ffffff; font-family:
Tahoma; font-size: 34pt; font-weight: bold; line-height: 30px; padding-
left: 2px; }
</style>
</HEAD>
<BODY>

<TABLE id="CustomControls" cellSpacing="0" cellPadding="0" width="100%"
bgColor="silver" border="1" height="100%" align="center">

<TR>
<TD class="tttDisplay" height="33%" width="100%" id="cctd1"
vAlign="center">
<maskededittextbox id="maskedEditTextBox1" height="20" width="136">
</maskededittextbox>
</TD>
</TR>

<TR>
<TD class="tttDisplay" height="33%" width="100%" id="cctd2"
vAlign="center">
<monthcalendar id="monthCalendar1" width="199"
height="155"></monthcalendar>
</TD>
</TR>

<TR>
<TD class="tttDisplay" height="33%" width="100%" id="cctd3"
vAlign="center">
<datagrid id="dataGrid1" width="304" height="144"></datagrid>
</TD>
</TR>

</TABLE>

</BODY>
</HTML>

```

The **CustomControlBase** implements the base functionality of the Windows forms control on the HTML tag element and loads the custom controls with its respective functionalities into the HTMLUI.

[C#]

```

// Event Handler Declaration.
this.htmluiControl1.PreRenderDocument += new
Syncfusion.Windows.Forms.HTMLUI.PreRenderDocumentEventHandler(this.htmlui
Control1_PreRenderDocument);

```

```

// Event that is to be raised when a tree of element has been created and
// their size and location have
// not been calculated yet.
private void htmluiControl1_PreRenderDocument(object sender,
Syncfusion.Windows.Forms.HTMLUI.PreRenderDocumentArgs e)
{
    Hashtable htmelements = new Hashtable();

    // Returns an hash of elements by their User Id.
    htmelements = e.Document.ElementsByUserID;
    BaseElement maskedEditTextBoxElement1 =
    htmelements["maskedEditTextBox1"] as BaseElement;

    // Here the base functionality of the 'this.maskedEditBox1' is
    // implemented to the 'maskedEditTextBoxElement1'.
    new CustomControlBase( maskedEditTextBoxElement1, this.maskedEditBox1 );
    BaseElement monthCalendarElement1 = htmelements["monthCalendar1"] as
    BaseElement;

    // Here the base functionality of the 'this.monthCalendar1' is
    // implemented to the 'monthCalendarElement1'.
    new CustomControlBase( monthCalendarElement1, this.monthCalendar1 );
    BaseElement dataGridElement1 = htmelements["dataGrid1"] as BaseElement;

    // Here the base functionality of the 'this.dataGrid1' is implemented
    // to the 'dataGridElement1'.
    new CustomControlBase( dataGridElement1, this.dataGrid1 );
}

```

[VB.NET]

```

Private Sub htmluiControl1_PreRenderDocument(ByVal sender As Object,
ByVal e As Syncfusion.Windows.Forms.HTMLUI.PreRenderDocumentArgs)
    Dim htmelements As Hashtable = New Hashtable()
    htmelements = e.Document.ElementsByUserID

    Dim maskedEditTextBoxElement1 As BaseElement = CType(If(OfType
    htmelements("maskedEditTextBox1") Is BaseElement,
    htmelements("maskedEditTextBox1"), Nothing), BaseElement)
    Dim oTemp1 As CustomControlBase = New
    CustomControlBase(maskedEditTextBoxElement1, Me.maskedEditBox1)

    Dim monthCalendarElement1 As BaseElement = CType(If(OfType
    htmelements("monthCalendar1") Is BaseElement,
    htmelements("monthCalendar1"), Nothing), BaseElement)
    Dim oTemp2 As CustomControlBase = New
    CustomControlBase(monthCalendarElement1, Me.monthCalendar1)

    Dim dataGridElement1 As BaseElement = CType(If(OfType
    htmelements("dataGrid1") Is BaseElement, htmelements("dataGrid1"),
    Nothing), BaseElement)

```

```
Dim oTemp3 As CustomControlBase = New CustomControlBase(dataGridElement1,
Me.dataGrid1)
End Sub
```

5.13 How To Load HTML Into the HTMLUI Control?

You can make use of the **GetControlByElement()** method of the **InputHTML** Interface to get an object for the control present in an HTML element in the HTMLUI control. If the HTML element does not contain any control in it, it returns a null value, by default.

[HTML]

```
<html>
  <body>
    <input type="text" id="txt"/>
  </body>
</html>
```

[C#]

```
//Initializing the respective control's object
private System.Windows.Forms.RadioButton htmlRadioButton;

private void htmluiControl1_LoadFinished(object sender, System.EventArgs e)
{
    //Collecting the html elements in a hashtable with their key as id
    Hashtable elements = this.htmluiControl1.Document.GetElementsByUserIdHash();
    BaseElement radioElem = (BaseElement)elements["radio1"];

    //Getting the Control from the html element and assigning it to the required object
    //The html element hereafter can be accessed with the help of the htmlRadioButton
    object
    htmlRadioButton = (RadioButton)
    this.htmluiControl1.Document.GetControlByElement(radioElem);
}
```

[VB.NET]

```
'Initializing the respective control's object
Private htmlRadioButton As System.Windows.Forms.RadioButton

Private Sub htmluiControl1_LoadFinished(ByVal sender As Object, ByVal e As
System.EventArgs)
'Collecting the html elements in a hashtable with their key as id
Dim elements As Hashtable = Me.htmluiControl1.Document.GetElementsByUserIdHash()
Dim radioElem As BaseElement = CType(elements("radio1"), BaseElement)
```

```
'Getting the Control from the html element and assigning it to the required object
'The html element hereafter can be accessed with the help of the htmlRadioButton
object
htmlRadioButton = CType(Me.htmluiControl1.Document.GetControlByElement(radioElem),
RadioButton)
End Sub
```

5.13.1 Loading As Startup Document

An HTML document can be loaded at startup by two ways:

- Using the Properties window
- By coding

Using the Properties window, involves specifying the location of the startup HTML file by using the **StartupDocument** property of the HTMLUI control. The link shown at the bottom of the Properties window can also be used for the same purpose.



Figure 57: HTMLUI control Property Grid

While using code for the Startup Document, it should be written in the **Form_Load** event that occurs before the form is displayed for the first time.

[C#]

```
// Get or Set the path to the Startup Document for the control.
private void Form1_Load(object sender, System.EventArgs e)
{
    this.htmluiControll.StartupDocument =
        @"C:\MyProjects\Startup\startup_page.htm";
}
```

[VB.NET]

```
' Get or Set the path to the Startup Document for the control.
Private Sub Form1_Load(ByVal sender As Object, ByVal e As System.EventArgs)
    Me.htmluiControll.StartupDocument = "C:\MyProjects\Startup\startup_page.htm"
End Sub
```

5.13.2 Loading At Run Time

HTML documents can also be loaded at run time. For example, in a file link where an HTML file may link to another file. In that case, a new file is loaded in the control after the one that was initially loaded.

The various ways of loading the document at run time from various resources are:

To load the file from disk into the HTMLUI, the following code snippet can be used.

[C#]

```
// Load the specified HTML document from Disk at runtime.
string filepath = @"C:\MyProjects\LoadHTML\FromDisk.htm";
this.htmluiControll.LoadHTML(filepath);
```

[VB.NET]

```
' Load the specified HTML document from Disk at runtime.
Private filepath As String = "C:\MyProjects\LoadHTML\FromDisk.htm"
Me.HtmluiControll.LoadHTML(filepath)
```

The above snippet can also be used to load HTML documents which are linked to the specified HTML documents, as links are easily invoked in HTMLUI.

To load a file from the URI, the following code snippet can be used.

[C#]

```
using System.Net;

Uri uri = new Uri( "http://www.syncfusion.com");
htmluiControl1.LoadHTML( uri );
```

[VB.NET]

```
Imports System.Net

Private uri As Uri = New Uri("http://www.syncfusion.com")
HtmluiControl1.LoadHTML(uri)
```

Loading HTML in the form of a string can be done as shown below.

[C#]

```
string htmlCode = "<HTML><HEAD><TITLE>HI</TITLE></HEAD>
<BODY bgcolor='#ffffff'><INPUT type='button' id='btn' /></INPUT></BODY></HTML>";
this.htmluiControl1.LoadFromString(htmlCode);
```

[VB.NET]

```
Private htmlCode As String = "<HTML><HEAD><TITLE>HI</TITLE></HEAD>
<BODY bgcolor='#ffffff'><INPUT type='button' id='btn' /></INPUT></BODY></HTML>"
Me.HtmluiControl1.LoadFromString(htmlCode)
```

To load a HTML file as an embedded resource:

1. Add an HTML file as an Embedded Resource to the application.
2. Set its **BuildAction** as Embedded Resource.
3. Include the following code snippet.

[C#]

```
Stream htmlStream =
(Stream)Assembly.GetExecutingAssembly().GetManifestResourceStream
("LoadingFileFromResource.resfile.htm");
this.htmluiControl1.LoadHTML(htmlStream);
```

[VB.NET]

```
Private htmlStream As Stream =
CType(System.Reflection.Assembly.GetExecutingAssembly().GetManifestResourceStre
am("LoadingFileFromResource.resfile.htm"), Stream)
Me.HtmluiControl1.LoadHTML(htmlStream)
```

 **Note:** The string entered inside the `GetManifestResourceStream` method is in reference to the Default namespace found in the Properties window of the C# file in the Solution Explorer. This may vary for the users.

5.14 How To Print the Contents Of the HTMLUI Control?

The document available in the HTMLUI control can be printed with the help of the **HTMLUIPrintDocument** class. The **Print** method of this class is used to start the document printing process.

[C#]

```
//represents printing support in the HTMLUI control
HTMLUIPrintDocument pd;
pd = new HTMLUIPrintDocument(this.htmluiControl1.Document);

// starts the printing process
pd.Print();
```

[VB.NET]

```
' represents printing support in the HTMLUI control
Private pd As HTMLUIPrintDocument
Private pd = New HTMLUIPrintDocument(Me.HtmluiControl1.Document)

' Starts the printing process
pd.Print()
```

5.15 How To Specify a CSS File For HTML Content?

Style sheets contain the styles to be applied for the elements in the HTML document. HTMLUI supports the use of styles in three modes:

- Inline style sheet (inside an HTML element)
- Internal style sheet (inside the tag)
- External style sheet (as a separate file)

The HTMLUI control supports formatting the HTML document with style sheets at run time. The **LoadCSS** method of the HTMLUI control loads the styles from the specified CSS and refreshes the current document on the HTMLUI control with the applied styles.

[C#]

```
// Loads styles from the specified CSS document from a System.IO.Stream and
refresh the current
// document using the styles
this.htmluiControl1.LoadCSS(@"C:\MyProjects\LoadCSS\style.css");
```

[VB.NET]

```
' Loads styles from the specified CSS document from a System.IO.Stream and
refresh the current
' document using the styles
Me.HtmluiControl1.LoadCSS("C:\MyProjects\LoadCSS\style.css")
```

5.16 How To Toggle the Visibility Of an HTML Element In the HTMLUI control At Run-time?

Each HTML element in the HTMLUI has an **xVisible** attribute by default that helps the user to toggle the visibility of a particular element. Since the xVisible is a bool property, the value to be set is either **True** or **False**, as string.

The following code snippet shows how the visibility of an element is toggled on the execution of an event.

[HTML]

```
<html>
  <body>
    <table>
      <tr>
        <td id="tdpopup">Cell Toggle</td>
        <td id="tdpopup">Cell Toggle</td>
      </tr>
      <tr>
        <td colspan="2">
          
        </td>
      </tr>
    </table>
  </body>
</html>
```


[C#]

```
IHTMLInputElement image = Global.Document.GetElementById("img");
image.Click += new EventHandler(image_Click);
IHTMLInputElement[] elem = this.htmluiControl1.Document.GetElementsByName("td");

public void image_Click(object sender, EventArgs e)
{
    string visibleString = "";
    htmluiControl1.BeginUpdate();
    if(this.bDescriptionHidden == false)
        visibleString = "false";
    else
        visibleString = "true";
    this.bDescriptionHidden = !this.bDescriptionHidden;
    foreach (IHTMLInputElement description in elem)
    {
        if(description.ID == "tdpopup")
            description.Attributes["xVisible"].Value = visibleString;
    }
    htmluiControl1.EndUpdate();
    this.htmluiControl1.Refresh();
}
```

[VB.NET]

```
Private image As IHTMLInputElement = Global.Document.GetElementById("img")
Private image.Click += New EventHandler(image_Click)
Private elem As IHTMLInputElement() =
Me.htmluiControl1.Document.GetElementsByName("td")

Public Sub image_Click(ByVal sender As Object, ByVal e As EventArgs)
    Dim visibleString As String = ""
    htmluiControl1.BeginUpdate()
    If Me.bDescriptionHidden = False Then
        visibleString = "false"
    Else
        visibleString = "true"
    End If
    Me.bDescriptionHidden = Not Me.bDescriptionHidden
    For Each description As IHTMLInputElement In elem
        If description.ID = "tdpopup" Then
            description.Attributes("xVisible").Value = visibleString
        End If
    Next description
    htmluiControl1.EndUpdate()
    Me.htmluiControl1.Refresh()
End Sub
```


Index

1

1 Overview 9

1.1 Introduction To Essential HTMLUI 9

1.2 Prerequisites and Compatibility 10

1.3 Documentation 11

2

2 Installation and Deployment 13

2.1 Installation 13

2.2 Sample and Location 13

2.3 Deployment Requirements 16

2.3.1 Toolbox Entries 16

2.3.2 DLLs 16

3

3 Getting Started 17

3.1 Lesson 1: Creating an HTML Display Application 18

3.1.1 Displaying HTML By Using the HTMLUI Control 18

3.2 Lesson 2: Creating an HTML Layout 21

3.2.1 Creating the User Interface 21

3.2.2 Creating And Displaying Custom Controls 25

4

4 Concepts And Features 30

4.1 Loading HTML 31

4.1.1 Loading As a Startup Document 31

4.1.1.1 Startup File Sample 33

4.1.2 Loading At Run Time 34

4.1.2.1 Loading the File From Disk 36

4.1.2.1.1 Load File From Disk Sample 37

4.1.2.2 As Links From One HTML Document To Another 37

4.1.2.2.1 File Links Sample 39

4.1.2.3 Loading the File From URI 40

4.1.2.4 Loading HTML Which Is In the Form Of Text 41

4.1.2.5 Load the File from Resource 41

4.1.2.5.1 Load Resource File Sample 45

4.10 HTML Tables 105

4.10.1 HTMLUI Tables Sample 106

4.11 HTML Layout 107

4.11.1 HTMLUI Chat Sample 108

4.12 HTML Renderer 108

4.12.1 HTMLUI Browser Sample 110

4.12.2 HTMLUI Editor Sample 111

4.13 Style Sheets CSS 112

4.13.1 Cascading Style Sheets 113

4.13.1.1 Inline StyleSheet 114

4.13.1.2 Internal StyleSheet 114

4.13.1.3 External StyleSheet 115

4.13.1.3.1 Design Time 116

4.13.1.3.2 Run Time 116

4.13.1.4 Class Selectors 118

4.13.1.4.1 Name Class Selectors 118

4.13.1.4.2 ID Class Selectors 118

4.13.1.5 CSS Comments 119

4.13.2 Supported CSS Attributes 120

4.13.2.1 Background - CSS 120

4.13.2.1.1 Background Image 120

4.13.2.1.2 Background Color 120

4.13.2.1.3 Background Repeat 121

4.13.2.2 Text – CSS 121

4.13.2.2.1 Text Color 121

4.13.2.2.2 Text Align 122

4.13.2.2.3 Text Decoration 122

4.13.2.3 Font – CSS 122

4.13.2.3.1 Font Family 123

4.13.2.3.2 Font Size 123

4.13.2.3.3 Font Style 123	4.22.1 HTMLUIScripting Sample 147
4.13.2.3.4 Font Weight 124	4.23 Text Selection 148
4.13.2.4 Border – CSS 124	4.23.1 HTMLUITextSelection Sample 149
4.13.2.4.1 Border Color 124	4.3 Control Events 47
4.13.2.4.2 Border Width 125	4.3.1 HTMLUI Control Events Sample 52
4.13.2.5 Padding – CSS 125	4.4 Element Events 53
4.13.2.5.1 Padding 126	4.4.1 HTMLUI Bubbling Events Sample 56
4.13.2.5.2 Padding Right 126	4.4.2 HTMLUI Element Events Sample 57
4.13.2.5.3 Padding Left 126	4.5 HTML Elements 58
4.13.2.5.4 Padding Top 126	4.5.1 Element Types 59
4.13.2.5.5 Padding Bottom 126	4.5.10 FONT Element 62
4.13.2.6 Dimension - CSS 126	4.5.11 FORM Element 62
4.13.2.7 Classification - CSS 127	4.5.12 H1 - H6 Header Elements 62
4.13.2.8 Positioning - CSS 127	4.5.13 HEAD Element 63
4.13.2.9 Pseudo - Classes 128	4.5.14 HTML Element 63
4.13.3 HTMLUIUseCSS Sample 128	4.5.15 HR - Horizontal Rule Element 63
4.13.3.1 HTMLUIElementsCSS Sample 129	4.5.16 I - Italics Element 63
4.14 HTMLUI Appearance 130	4.5.17 IMG - Image Element 63
4.14.1 HTMLUIAppearance Sample 131	4.5.18 INPUT Element 64
4.15 HTML Format 132	4.5.19 LI - List Element 65
4.15.1 HTMLFormat Sample 133	4.5.2 A - Anchor Element 59
4.16 Element Format 134	4.5.20 LINK Element 65
4.16.1 ElementFormat Sample 136	4.5.21 OL - Ordered List Element 66
4.17 Exporting 137	4.5.22 P - Paragraph Element 66
4.17.1 HTMLUIExporting Sample 139	4.5.23 PRE - Preformatted Element 66
4.18 Keyboard 139	4.5.24 SCRIPT Element 66
4.18.1 HTMLUIKeyboard Sample 140	4.5.25 SELECT Element 67
4.19 Printing 140	4.5.26 SPAN Element 68
4.19.1 HTMLUIPrinting Sample 143	4.5.27 STRONG Element 68
4.2 MHT Formats 45	4.5.28 STYLE Element 68
4.20 Searching 143	4.5.29 TABLE Element 69
4.20.1 HTMLUISearching Sample 144	4.5.3 B - Bold Element 60
4.21 Scrolling 145	4.5.30 TD - Table cell Element 69
4.21.1 HTMLUIAutoScroll Sample 146	4.5.31 TEXTAREA Element 70
4.22 Scripting 146	4.5.32 TH - Table Head Element 70

4.5.33 TR - Table Row Element	70	4.6.32 TITLE - Title Tag	95
4.5.34 UL - Unordered List Element	71	4.6.33 TR - Table Row Tag	96
4.5.35 U - Underline Element	71	4.6.34 UL - UnOrdered List Tag	96
4.5.4 BODY Element	60	4.6.35 HTML Tags Sample	97
4.5.5 BR - Break Element	61	4.6.4 Comment Tag	73
4.5.6 CODE Element	61	4.6.5 Font Style Tags	74
4.5.7 CUSTOM Element	61	4.6.6 BODY - Body Tag	75
4.5.8 DIV - Division Element	62	4.6.7 BR - Break Tag	75
4.5.9 EM - Emphasize element	62	4.6.8 DIV - Division Tag	76
4.6 HTML Tags	71	4.6.9 FORM - Form Tag	76
4.6.1 A - Anchor Tag	72	4.7 Custom Controls	98
4.6.10 HEAD - Head Tag	77	4.7.1 Custom Controls Sample	100
4.6.11 H1 - H6 Header Tag	78	4.8 HTML Forms	101
4.6.12 HR - Horizontal Rule Tag	78	4.8.1 HTMLUIForms Sample	102
4.6.13 HTML - HTML Tag	79	4.9 HTML Bookmarks	103
4.6.14 IMG - Image Tag	80	4.9.1 HTMLUI Bookmarks Sample	104
4.6.15 INPUT - Input Tag	80	5	
4.6.16 LI - List Item Tag	82	5 Frequently Asked Questions	151
4.6.17 LINK - Link Tag	82	5.1 How To Access All the Child Elements Of an HTML Element In the HTMLUI Control?	151
4.6.18 OL - Ordered List Tag	83	5.10 How To Enable User Interaction With the HTML Elements	164
4.6.19 OPTION - Option Tag	84	5.11 How To Get an Object For the Control Present In an HTML Element In the HTMLUI Control?	167
4.6.2 ABBR - Abbreviation Tag	72	5.12 How To Load Custom Controls As Part Of the HTML Layout?	168
4.6.20 P - Paragraph Tag	84	5.13 How To Load HTML Into the HTMLUI Control?	171
4.6.21 PRE - Preformatted Tag	85	5.13.1 Loading As Startup Document	172
4.6.22 SCRIPT - Script Tag	86	5.13.2 Loading At Run Time	173
4.6.23 SELECT - Select Tag	87	5.14 How To Print the Contents Of the HTMLUI Control?	175
4.6.24 SPAN - Span Tag	88	5.15 How To Specify a CSS File For HTML Content?	175
4.6.25 STYLE - Style Tag	88	5.16 How To Toggle the Visibility Of an HTML Element In the HTMLUI control At Run-time?	176
4.6.26 SUB - Subscript Tag	90		
4.6.27 SUP - Superscript Tag	90		
4.6.28 TABLE - Table Tag	91		
4.6.29 TD - Table Cell Tag	92		
4.6.3 Acronym Tag	73		
4.6.30 TEXTAREA - Text Area Tag	93		
4.6.31 TH - Table Header Tag	94		

- 5.2 How To Access the HTML Elements Loaded Into the Control? 153
- 5.3 How To Access the Inner HTML Text Of the Current HTML Element In the HTMLUI Control? 154
- 5.4 How To Access the Name Of an HTML Element At Run-time? 155
- 5.5 How To Add an Attribute To an HTML Element And Change Its Value At Run-time? 156
- 5.6 How To Change a Characteristic Of an HTML Element Before It Is Displayed? 157
- 5.7 How To Change the Default Font Used For Rendering the HTML Document In the HTMLUI Control? 160
- 5.8 How To Delete an HTML Element From a Document Loaded Into the HTMLUI Control At Run-time? 161
- 5.9 How To Enable the HTMLUI Control To Load HTML Documents That Have Been Dragged Onto the Control? 162