Software Systems

WebTemplate

Development Kit 1.0

Support

Web : www.software-systems.de
e-mail : support@software-systems.de

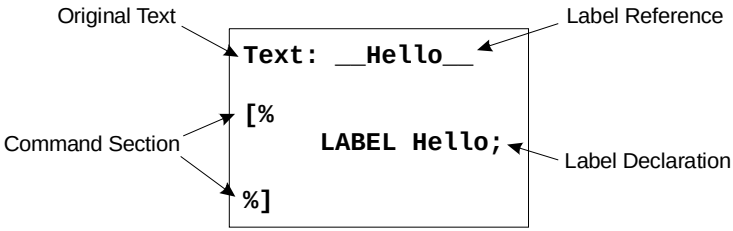
TABLE OF CONTENT

1 Introduction

WebTemplate is a universal solution for template management, controling and processing. It is useful for automatic HTML generation, e-mail generation, mail merge or other similar purposes. WebTemplate works with many popular development environments like Visual Basic, CA-Visual Objects, Access, Delphi, etc..

WebTemplate is implemented as an ActiveX Automation Server and uses the ProgIDs "WebTemplate.Template" and "WebTemplate.TLabel". Please look at the README.TXT file for installation instructions. With the WebTemplate API (Chapter 3), the developer can load templates, retrieve template, label and variable information, insert data and create the output.

The picture below shows a simple template file.



A template consists of original (raw) text, label references and a command section. A label reference is a placeholder and will be replaced with your text at output creation. Another template element is the command section with statements that further describes the template or label behaviour. The command section is also used to insert management information, to declare label objects, or to create variables. The command section is completely removed from the processed template.

The following steps are necessary to handle a template:

- Create template object
- Load template file
- Query template and label information
- Add label data
- Create output

Three kinds of data are associated with a label object: user data, default data and include file data. It's not possible to directly assign data to label references, instead the user assigns data to label objects. WebTemplate uses the following processing scheme to select the right label data for insertion.

```
IF Label.UseIncludeFile THEN
   Insert include file data
ELSE
   IF Label.UserDataSize > 0 THEN
      Insert user data
   ELSE
      Insert default data
   END
END
```

Quick start: Look at the examples chapter with template examples and Visual Basic code. Try them out and check the Template.Error and Template.ErrorExtra property for detailed information if something goes wrong.

2 Template Format

2.1 Label Reference

Label references are placeholders for label data and consist of a label name with two leading and trailing underscores. WebTemplate will implicitly create a label object, if it doesn't exist.

Syntax: _ _*label-name*_ _

2.2 Command Section

A command section is used for setting template properties, declaring label objects, setting label object properties and creating variables. Zero or more command sections are allowed in a template.

A command section starts with **[%** and ends with **%]**. It may contain zero or more statements and every statement must end with a semicolon. Spaces, tabs and line feeds are ignored. Example:

```
[%
    Template.Id = 123;
    Template.Description = "Sample Template";
    LABEL MyFirstLabel;
%]
```

Note that if you create a template with a HTML layout editor, insert the command section in a HTML comment.

WebTemplate supports the data types integer (e.g. 123; -123), real number (e.g. 1.2; -1.2) and string. A string is a series of characters enclosed in double or single quotes. Strings may contain the opposite quote as data:

```
"It's easy", 'Keep "cool"'
```

A string can span several lines:

```
"multiline
text"
```

If WebTemplate detects a multiline string it always normalizes the line break(s) and returns a CRLF pair as line separator. WebTemplate supports string concatenation, two or more adjacent strings will be concatenated into a single string:

```
"I'm" "one" 'string'
```

is concatenated to

```
"I'm one string"
```

Strings may also contain the following escape character sequences.

Escape	Character
`\\`	$5C_{16}$, Backslash
`\n`	$0A_{16}$, LF
`\r`	$0D_{16}$, CR
`\t`	09_{16}, TAB
`\'`	Single Quote
`\"`	Double Quote
\xHH	Hexadecimal Code

2.2.1 Variables

Variables can be used by the developer to place additional information into a template file. Variable names may contain letters, digits, but must start with a letter. Variable names are case-sensitive.

Syntax: *variable-name* = *value*;

Examples: `MaxRows = 10; BackGround = "Yellow";`

2.2.2 Template Object

There's only one template object per template file. WebTemplate offers some template properties that can be used to set management information. The current template properties do not affect the output or workflow. Property names are case-sensitive.

Syntax: `Template.`*property-name* = *value*`;`

The following properties are available:

`Template.Description`
Sets the template description.

Value
Any value.

`Template.Id`
Sets the template identifier.
Value
Any value.

`Template.Language`
Sets the template language identifier.
Value
Any value.

`Template.Name`
Sets the template name.
Value
Any value.

`Template.Version`
Sets template version information.
Value
Any value.

2.2.3 Label Objects

A template can contain zero or more label objects. WebTemplate explicitly creates label objects with the LABEL declaration command or implicitly with a label reference. The LABEL command sets the object's declared flag. The LABEL command has several enhancements that automatically encodes and assigns default data or binds an include file with the appropriate encoding option. Label names may contain letters, digits, but must start with a letter. Label names are case-sensitive.

The following syntax creates a label object.

Syntax: `LABEL` *label-name*;

Example: `LABEL MyLabel;`

The following syntax creates a label object, optionally encodes *string-value* and assigns the result as default data.

Syntax: `LABEL` *label-name* `DEFAULT` *string-value*
[`OPTION` (`NOENCODING` | `URLENCODING` | `HTMLENCODING`)] `;`

Examples:
```
LABEL MyLabel DEFAULT "Hi!";
LABEL MyLabel DEFAULT "Hi!" OPTION URLENCODING;
```

The following syntax creates a label object, associates an include file and an optional include file encoding.

Syntax: `LABEL` *label-name* `INCLUDE` *string-value*
[`OPTION` (`NOENCODING` | `URLENCODING` | `HTMLENCODING`)] `;`

Examples:
```
LABEL MyLabel INCLUDE "stat.dat";
LABEL MyLabel INCLUDE "stat.dat" OPTION
HTMLENCODING;
```

Label properties control the label processing and hold management information.

General Syntax: *label-name*`.`*property-name* `=` *value*`;`

The following properties are available:

label-name`.DefaultData`
Sets the default data.
Value
Any value.

label-name`.DefaultUserDataEncoding`
Sets the default user data encoding. This property controls the behaviour of the AddUserData method.
Value
`NOENCODING`, `URLENCODING` or `HTMLENCODING`

label-name`.Description`
Sets the label description
Value
Any value.

label-name.IncludeFileEncoding
Sets the encoding of the include file.
Value
NOENCODING, URLENCODING or HTMLENCODING

label-name.IncludeFileName
Sets the file to include. Note that \ within the file name must be written as \\.
Value
Must be a *string-value.*

label-name.IncludeFileRefresh
Sets a boolean value indicating the current include file loading. TRUE, the include file is always loaded before writting to the output stream. FALSE, the include file is loaded once.
Value
TRUE or FALSE

label-name.UseIncludeFile
Enables or disables file inclusion.
Value
TRUE or FALSE

2.2.4 Comments

A command section can contain single-line and multiline comments. Single-line comments start with //, anything after the two slashes up to the end of the line is ignored. Multiline comments start with /* and and end with */, anything between /* and */ is ignored.

2.3 Reserved Words

The following words have a special meaning and cannot be used as label, variable or property names: DEFAULT, FALSE, HTMLENCODING, INCLUDE, LABEL, NOENCODING, OPTION, Template, TRUE, URLENCODING.

2.4 Template Examples

Example 1

Template: SimpleTemplate.htm

```
<HTML>
   <HEAD>
     <TITLE>Simple Template Example</TITLE>
   </HEAD>
   <BODY>

   __SimpleLabel__

   </BODY>
</HTML>
```

VB Code:

```
Dim oTemplate    As New WebTemplate.Template
Dim oSimpleLabel As WebTemplate.TLabel

oTemplate.Initialize "", "", ""

oTemplate.LoadTemplateFile "SimpleTemplate.htm", 0

Set oSimpleLabel = oTemplate.GetLabel( "SimpleLabel" )
oSimpleLabel.AddUserData "I'm a simple text!", _
                   ST_ENCODING_HTML

oTemplate.CreateOutputFile "SimpleOutput.htm", _
                   ST_OUTPUT_OVERWRITE
```

Output: SimpleOutput.htm

```
<HTML>
   <HEAD>
     <TITLE>Simple Template Example</TITLE>
   </HEAD>
   <BODY>

   I’m a simple text!

   </BODY>
</HTML>
```

Example 2

Template: Hello.htm

```
<HTML>
   <HEAD>
     <TITLE>Hello World Example</TITLE>
   </HEAD>
   <BODY>

   __Hello__

   </BODY>
   <!--
     [%
        LABEL Hello
            DEFAULT "Sorry! No text."
            OPTION HTMLENCODING;
     %]
     -->
</HTML>
```

VB Code:

```
Dim oTemplate    As New WebTemplate.Template
Dim oHelloLabel  As WebTemplate.TLabel

oTemplate.Initialize "", "", ""

oTemplate.LoadTemplateFile "Hello.htm", 0

Set oHelloLabel = oTemplate.GetLabel( "HelloLabel" )
oHelloLabel.AddUserData "Hello World!", ST_ENCODING_HTML

oTemplate.CreateOutputFile "HelloOutput.htm", _
                  ST_OUTPUT_OVERWRITE
```

Output I: HelloOutput.htm

```
<HTML>
   <HEAD>
     <TITLE>Hello World Example</TITLE>
   </HEAD>
   <BODY>

   Hello World!

   </BODY>
   <!--
     -->
</HTML>
```

Output II: HelloOutput.htm (with removed line `oLSimple.AddUserData`)

```
<HTML>
   <HEAD>
     <TITLE>Hello World Example</TITLE>
   </HEAD>
   <BODY>

   Sorry! No text.

   </BODY>
</HTML>
```

Example 3

Template: SalesTemplate.htm

```
<HTML>
   <HEAD>
     <TITLE>__SalesStatTitle__</TITLE>
   </HEAD>
   <BODY>
   <PRE>
__SalesStatText__
   </PRE>
   </BODY>
<!-- [%
        LABEL SalesStatTitle
            DEFAULT "Sales Statistic"
            OPTION HTMLENCODING;
        LABEL SalesStatText
            INCLUDE "SalesData.txt"
            OPTION HTMLENCODING;
     %] -->

</HTML>
```

VB Code:

```
Dim oTemplate As New WebTemplate.Template

oTemplate.Initialize "", "", ""

oTemplate.LoadTemplateFile "SalesTemplate.htm", 0

oTemplate.CreateOutputFile "SalesStatOutput.htm", _
                    ST_OUTPUT_OVERWRITE
```

Output: SalesStatOutput.htm

```
<HTML>
   <HEAD>
     <TITLE>Sales Statistic</TITLE>
   </HEAD>
   <BODY>
     <PRE>
        ...
        .... Content of file SalesData.txt comes here
        ...
     </PRE>
   </BODY>
<!--  -->
</HTML>
```

Example 4

Template: AdvertTemplate.htm

```
<HTML>

.....

   <A HREF="__Advert1URL__" ALT="__Advert1Text__">
      <IMG SRC="__Advert1Image__" ALT="__Advert1ImageText__">
   </A>

.....

<!-- [%

         // some management information
         Template.Name = "Advertising Example";
         Template.Version = 1.0;
         Template.Language = "eng";
         Template.Id = 42;

         // declare labels
         LABEL Advert1URL;
         LABEL Advert1Text
         LABEL Advert1Image;
         LABEL Advert1ImageText;

         Advert1Image.DefaultData = "NULL.GIF";

%] -->
....

</HTML>
```

VB Code:

```
Dim oTemplate As New WebTemplate.Template
Dim oAdvert1URL As WebTemplate.TLabel
Dim oAdvert1Text As WebTemplate.TLabel
Dim oAdvert1Image As WebTemplate.TLabel
Dim oAdvert1ImageText As WebTemplate.TLabel

oTemplate.Initialize "", "", ""

oTemplate.LoadTemplateFile "AdvertTemplate.htm", 0

'retrieve management information
'   sName     = oTemplate.Name
'   sVersion  = oTemplate.Version
'   sLanguage = oTemplate.Language
'   sId       = oTemplate.Id

Set oAdvert1Image = oTemplate.GetLabel( "Advert1Image" )
If Not (oAdvert1Image Is Nothing) Then
   oAdvert1Image.AddUserData "http://www.ex.com/advert.gif",_
                     ST_ENCODING_NO
End If

Set oAdvert1URL = oTemplate.GetLabel( "Advert1URL" )
If Not (oAdvert1URL Is Nothing) Then
   oAdvert1URL.AddUserData "http://www.ex.com/", _
                     ST_ENCODING_NO
End If


oTemplate.CreateOutputFile "AdvertOutput.htm", _
                     ST_OUTPUT_OVERWRITE
```

3 ActiveX Automation API

3.1 Template Object

3.1.1 Properties

Description

Returns a string that optionally holds a user-defined template description.

Template file entry: `Template.Description =` *value*

Error

Returns the last error as a long value.

ErrorExtra

Returns a string that contains one or more lines with an error history. The first line represents the last error. Every line contains an error or warning entry starting with a decimal result number (STRES_...) optionally followed by a single space character and a description. Lines are separated with CRLF.

Example:

```
24 Line:21 Column:15 HelloLabelII.Abc↵
25 Line:13 Column:11 Abc
```

Id

Returns a string that optionally contains user-defined identification information.

Template file entry: `Template.Id =` *value*

Language

Returns a string that optionally contains user-defined language information.

Template file entry: `Template.Language =` *value*

Name
Returns a string that optionally contains user-defined name information.

Template file entry: `Template.Name = value`

NumberOfLabels
Returns a long value that represents the number of labels within the current template.

NumberOfVariables
Returns a long value that represents the number of variables within the current template.

TemplateDirectory [= *directory*]
Returns or sets the template home directory. The template processor uses this value only if the LoadTemplateFile method will be called with a relative file path for the template file. The parameter *directory* is a string expression.

TemplateFile
Returns a string expression that holds the file name of the current template.

Version
Returns a string that optionally contains user-defined version information.

Template file entry: `Template.Version = value`

Warnings
Returns a boolean value. True, warnings detected and false, no warnings. Please check the property ErrorExtra for the actual warning text.

3.1.2 Methods

Clear( )
This method clears the current template and frees any used memory.

ClearError( )
This method clears all error information used by the template.

ClearUserData( )

This method clears the user data portion of all labels

CreateOutputFile(*fileName, outputOptions*)

The CreateOutputFile method generates the final result file.

Parameters

fileName

Name of the destination file that receives the final data.

outputOptions

Symbol	Description
ST_OUTPUT_KEEP	Don't overwrite an existing file.
ST_OUTPUT_OVERWRITE	Overwrite existing file.

Return

Result code indicating success or failure.

CreateOutputString(*outputOptions*)

The CreateOutput method generates the final result as a string value.

Parameters

outputOptions

This parameter is reserved for future use and must be 0.

Return

String containing the processed template output.

EnumLabelNames(*nEnum*)

The EnumLabelNames method enumerates the labels within a template. The numerical parameter *nEnum* can range from 1 to NumberOfLabels.

Parameters

nEnum

Counter value

Return

String containing the label name. If empty, stop enumeration.

EnumVariableNames(*nEnum*)

The EnumVariableNames method enumerates the variables within a template. The numerical parameter *nEnum* can range from 1 to NumberOfVariables.

Parameters

nEnum

Counter value

Return

String containing the variable name. If empty, stop enumeration.

GetLabel(*labelName*)

This method returns a reference to a label object. The label is identified through the parameter *labelName*.

Parameters

labelName

Name of label

Return

Reference to TLabel object.

GetVariableValue(*variableName*)

The GetVariableValue method returns the assigned value as text.

Parameters

variableName

Name of variable

Return

String containing the assigned value.

GetVariableValueType(*variableName*)

The GetVariableValueType method returns the value type of the assigned data.

Parameters

variableName

Name of variable

Return

Value type constant. It can be one of the following: ST_VTYPE_NULL, ST_VTYPE_INT, ST_VTYPE_REAL, ST_VTYPE_STRING.

HTMLEncode(*text*)

The HTMLEncode method applies HTML encoding to the specified string.

Parameters

text

String to convert

Return

String containing the encoded text

Initialize(*sn, keyHigh, keyLow*)

The Init method initializes the current template object. Call it after creating a new template object and before using any other template method.

Parameters

sn

String that contains the product serial number.

keyHigh

String that contains the high part of the activation key.

keyLow

String that contains the low part of the activation key.

Return

Result code indicating success or failure.

IsLabel(*labelName*)

The IsLabel method checks whether a certain label exist.

Parameters

labelName:

Label name to look for.

Return

Boolean value indicating failure or success.

IsVariable(*variableName*)

The IsVariable method checks whether a certain variable exist.

Parameters

variableName

Variable name to look for.

Return

Boolean value indicating failure or success.

LoadTemplateFile(fileName, loadOptions)

The LoadTemplateFile method loads and analyses a template file.

Parameters

fileName

Name of template file to load.

loadOptions

Currently not used, set it to 0.

Return

Result code indicating success or failure.

URLEncode(*text*)

The URLEncode method applies URL encoding to the specified string.

Parameters

text

String to convert.

Return

String containing the encoded text.

3.2 TLabel Object

3.2.1 Properties

Declared

Returns a boolean value. True, the label is declared in the template and false, the template contains one or more label references without declaration.

DefaultUserDataEncoding

Returns or sets the default encoding option for the user data part. This property controls the behaviour of the AddUserData method. If AddUserData is called with an encodingOption of ST_ENCODE_DEFAULT, the setting of DefaultUserDataEncoding will take affect. Valid property values are:

Symbol	Description
ST_ENCODING_NO	Do not encode text
ST_ENCODING_URL	Convert text into URL format
ST_ENCODING_HTML	Convert text into HTML format

Template file entry: *label*`.DefaultUserDataEncoding` = *value*

Description

Returns a string that optionally holds an user-defined description.

Template file entry: *label*`.Description = ` *value*

IncludeFileDirectory

Returns or sets a string that optionally holds an absolute directory path for the include file.

IncludeFileEncoding

Returns or sets the include file encoding. Valid numerical property values are ST_ENCODING_NO, ST_ENCODING_URL and ST_ENCODING_HTML.

Template file entry: *label*`.IncludeFileEncoding` = *value*

IncludeFileName

Returns or sets a string that holds the name of the file to include. If the property contains a relative file specification, the property IncludeFileDirectory must be set. The name is concatenated with the property IncludeFileDirectory if a relative file specification is used.

Template file entry: *label*`.IncludeFileName =` *string-value*

IncludeFileRefresh

Returns or sets a boolean value indicating the current include file loading. True, CreateOutputFile and CreateOutputString always load the include file data before written into the output stream. False, the include file will be loaded once.

Template file entry: *label*`.IncludeFileRefresh =` *boolean-value*

Name

Returns a string that holds the label name.
Template file entry: *label*`.Name =` *string-value*

NumberOfReferences

Returns a numerical value representing the number of label references in the template.

UseIncludeFile

Returns a boolean value. True, the include file data will be inserted in the output and false, the user data should be used.

Template file entry: *label*`.UseIncludeFile =` *boolean-value*

3.2.2 Methods

AddUserData(*text, encodingOptions*)

The AddUserData method adds text to the user data. The content of *text* is encoded as specified through the argument *encodingOptions* and then appended to the current content.

Template file entry: *label*`.DefaultData =` *string-value* or
`LABEL` *label-name* `DEFAULT` *string-value*

Parameters

text

String data that should be added.

encodingOptions

Numerical constant which can be one of the following:

Symbol	Description
ST_ENCODING_DEFAULT	Encode as specified in the template file
ST_ENCODING_NO	Do not encode *text*
ST_ENCODING_URL	Convert *text* into URL format
ST_ENCODING_HTML	Convert *text* into HTML format

Return

Result code indicating success or failure.

ClearUserData( )

Clears the user data.

GetDefaultData( )

The GetDefaultData method returns the assigned default data.

SetDefaultData(*text, encodingOptions*)

The SetDefaultData method sets the default data portion. Any previously contained default data is replaced by the new data.

Parameters

text

String data that should be the new default data.

encodingOptions

Numerical constant which can be one of the following:

Symbol	Description
ST_ENCODING_NO	Do not encode *text*
ST_ENCODING_URL	Convert *text* into URL format
ST_ENCODING_HTML	Convert *text* into HTML format

Return

Result code indicating success or failure.

4 Constants

Result Codes

STRES_SUCCESS	0
STRES_INCFILE_NOT_FOUND	1
STRES_INCFILE_OPEN_ERROR	2
STRES_INCFILE_READ_ERROR	3
STRES_INVALID_BUFFER_SIZE	4
STRES_INVALID_DATA_TYPE	5
STRES_INVALID_HANDLE	6
STRES_INVALID_PARAM	7
STRES_INVALID_PATH	8
STRES_LABEL_NOT_FOUND	9
STRES_NO_TEMPLATE_LOADED	10
STRES_NOT_ENOUGH_MEMORY	11
STRES_NOT_INITIALIZED	12
STRES_NOT_LICENSED	13
STRES_OUTPUT_FILE_ALREADY_EXIST	14
STRES_OUTPUT_WRITE_ERROR	15
STRES_TEMPLATE_NOT_FOUND	16
STRES_TEMPLATE_OPEN_ERROR	17
STRES_TEMPLATE_FORMAT_ERROR	18
STRES_TEMPLATE_EMPTY	19
STRES_UNEXPECTED_EOF	20
STRES_UNKNOWN_EXCEPTION	21
STRES_USER_CANCEL	22
STRES_VARIABLE_NOT_FOUND	23

Value Types

ST_VTYPE_NULL	0
ST_VTYPE_INT	1
ST_VTYPE_REAL	2
ST_VTYPE_STRING	3

Encoding Options

ST_ENCODING_DEFAULT	-1
ST_ENCODING_NO	0
ST_ENCODING_URL	1
ST_ENCODING_HTML	2

Create Options

ST_OUTPUT_OVERWRITE	0
ST_OUTPUT_KEEP	1