WS Functional Automation

8.16 wsprint

Purpose

With wsprint, You can use the GuiXT WS platform to print directly from WS scripts. The WS platform extends the functionality of WS by providing the ability to use printers through Webscript. The wsprint.dll contains a set of libraries that you can use to access and use printers from within GuiXT WS scripts. Currently, this functionality is in use in the field by several of Synactive's customers, most notably COMPANY_NAME. The wsprint functionality is available for use with the following Synactive solutions:

- Server-based Implementation (GuiXT Server)
- Web-based Implementation (Web Server)
- Local Implementation (Liquid UI for SAP GUI)
- Offline implementation

The wsprint functionality can be used with Synactive's Offline solution (for Offline 3.1.140.0 or later releases) in addition to the local and Server-based implementations already mentioned. This highlights one of the great strengths of WS - you can write a single script and implement it with multiple touchpoints.

To use wsprint, you must first load the necessary files using the load command. These files are as follows.

- wsprint.dll
- wsprint.js

Note: This file is only necessary if you are using the Javascript functions.

If the wsprint.dll is in the home directory, the syntax to load the files is as follows.

```
load(‘wsprint’);
```

If the files are not in the home directory, you will need to specify the full path as shown below.

```
load('C:\\offline\\wsprint');
```

Initializing the wsprint Object To initialize the wsprint object, perform the configuration required to print and to access printers, use the following code.

```
var a = new Printer();
```

To print to a local printer, you can use the following syntax.

```
var a = new Printer("printername");
```

If you are using a network printer, please note that you must supply the full path to the printer, as in the following example.

```
var a = new Printer("\\server\printername");
```

WSPrint Functions

List of functions available with wsprint.

There are several functions available through wsprint. You must access all of these functions through an instance of the 'Printer' class, by calling the print() function on that instance. The text that you want to print is passed to the function as an argument. All other features of the wsprint class are controlled through a parameter that contains a list of the options and the values between the parentheses. A list of the available functions is below.

Constructor

The Constructor function initializes the PrinterService object, tries to access the printer and performs the setup required to print. To print to the system's default printer, use the following syntax.

```
a= new Printer();
```

If you want to print to a printer that is not the system default, please use the following syntax:

```
a= new Printer('printername');
```

Note: If you are printing to a network printer, you must use the full path of the printer.

Print

This function is a method of the print object and passes a string to the print() function, which then prints the line. The syntax is as follows.

```
a.print(‘TEST’);
```

The resulting output is left-aligned by default, but you can use the xPos option to re-align your text as shown below.

```
a.print({xPos:100},'TEST');
```

You can also use the printCenterAlign() or the PrnitRightAlign() functions. The syntax for each is shown below.

```
printCenterAlign(string); printRightAlign(string);
```

To insert line breaks, use the \n character. An example is shown below.

```
a.print('\nTEST 1');
```

Print Linestyle

To print a horizontal line, provide the ‘lineStyle’ option with the type of line to draw. An example is shown below.

```
a.print({lineStyle:'DASH'});
```

The option we used is the 'DASH' option, which results in a line broken into dashes. The options for the linestyle option are as follows.

- NORMAL: Prints a solid line.
- DASH: Prints a dashed line.
- DOT: Prints a dotted line.

Move Print to New Page

To move printing to a new page, use the NEWPAGE option command. Please note that any time you use an option command, you must specify it with the keyword 'cmd' as shown in the example. The syntax is shown below:

```
a.print({cmd:’NEWPAGE’});
```

End Printing

Use the FLUSH option command to do the following.

- End printing
- Send the job to the printer

The syntax is shown below:

```
a.print({cmd:’FLUSH’});
```

Line Spacing

Use the spacing option to set the space between lines of text. The value is numerical and the syntax is as follows:

```
a.print({spacing:20});
```

Set Margins

Use the margin option to specify the page margins. You can create an array and assign the array name to the margin option as shown in the following examples. The first example demonstrates assigning margins manually. The second example demonstrates assigning margins by using an array.

```
var iTOP = 20;
var iBOTTOM = 20;
var iLEFT = 20; var iRIGHT = 20;
var pMargin = [iTOP, iBOTTOM, iLEFT, iRIGHT];
a.print({margin:pMargin});
```

Note: To leave a parameter unchanged from the default, set it to '-1'.

Set Font

Use the font option to specify the type of font. Within the font, you can create a list of font options as shown in the example below.

```
var aFont={type:"Times New Roman", size:15, color:"Blue", italic:true,
 underline:true};
a.print({font:aFont});
```

Set Font Color

As introduced above, you can specify the font options in the font option. To specify colors, you either use the color name as in the previous example, or you can use the RGB values. The color names supported in wsprint are as follows.

- Aqua
- Azure
- Beige
- Black Blue
- Brown
- Crimson
- Cyan
- Gray
- Light Green
- Navy
- Red
- Violet

If you use RGB values to specify the colors, the syntax is as follows.

```
var iRcolor=200;
var iGcolor=210;
var iBcolor=100;
var color = [iRcolor,iGcolor,iBcolor];
var aFont={ color:zColor};
```

Get Printable Page Width

Use the printWidth member of the Print object. This will return the printable width of the page in pixels. Please note that the return value offsets the left and right margins from the page width. The syntax is as follows.

```
page_width=a.printWidth;
```

Get Text Width in Pixels

Use the bPrint:false option to get the width of some specified text in pixels. This will get the width in the currently specified font. The syntax is as follows:

```
text_width=a.print({bPrint:false},str);
```

Javascript Extensions for WSPrint

The extensions for WSPrint available with Javascript. In addition to the main wsprint.dll, Synactive also provides a Javascript extension called wsprint.js. This file is a Javascript extension and contains some additional functions for the WSPRint functionality. These functions are explained below.

tablePrint

Use the tablePrint() function to print text in a table format. You will specify the contents of the columns in one row as array, and then create an array of the rows that you wish to print. Once the array is created, you can call the tablePrint() function on the array. An example is shown below.

```
var row1=["Row1 Col1","Row1 Col2"];
var row2=["Row2 Col1","Row2 Col2"];
var table=[row1,row2];
a.tablePrint(1);
```

printCenterAlign

We introduced the printCenterAlign() function previously. The text to be printed is passed to the function as a string. An example is shown below.

```
a.printCenterAlign('MYMART');
```

printRightAlign

The printRightAlign() function works very similarly to the printCenterAlign() function above. The text is passed as a string and the result will be aligned to the right. An example is shown below.

```
a.printRightAlign('Right Align Text');
```

WSPrint Example

A real world example demonstrating the wsprint functionality. In this example, we will use wsprint to perform a number of printing operations. These examples are for the Offline solution, but the scripts can be used on any of the supported interfaces.

The files are as follows

- align.js
- print.js
- printreceipt.js

- table.js

Note: In all of the examples, the wsprint.js file resides in the same directory as the scripts, so we do not need to specify the full path of the file. In addition, these examples are all written as try-catch statements This is not required, but we highly recommend including error handling so that you can more easily troubleshoot if anything does not work.

Aligning Text

A real world example demonstrating the aligning functionality in WSPrint In this example, we will align some test before we print it. The process is as follows.

1. Define the printer that we will use. In this case, we will use the built-in XPS DOcument Writer.
2. Once the printer is defined, we will specify the margins and place them into an array.
3. Specify the output font and size.

In this example, we chose Times New Roman and set the size to 15. The script is shown below.

```
System.LoadFile('wsprint.js');
 function printAlignTest() {
 var a, b;
 try {
 a = new Printer('Microsoft XPS Document Writer');

 var iTOP = 20;
 var iBOTTOM = 20;
 var iLEFT = 20;
 var iRIGHT = 20;
 var pMargin = [iTOP, iBOTTOM, iLEFT, iRIGHT];
 var aFont={type:"Times New Roman", size:15};

 a.print({font:aFont,margin:pMargin});
 a.printCenterAlign('Center Align Text\n');
 a.printRightAlign('Right Align Text');
 }
 catch(err){
 System.TraceOutput(err);
 }
 }
printAlignTest();
```

Printing

WS Functional Automation

A real world example demonstrating the wsprint functionality. In this example, we will align some test before we print it. The process is as follows.

1. Define the printer that we will use. In this case, we will use the built-in XPS DOcument Writer.
2. Once the printer is defined, we will specify the margins and place them into an array.
3. Specify the output font and size. In this example, we chose Times New Roman and set the size to 15. We added the underline style as well.
4. Call the print() function, passing the margin array and the font information as options.
5. Verify that the print operation succeeded.

The script is shown below.

```
function printTest() {
 var a, b;
 try {
 a = new Printer('Microsoft XPS Document Writer');

 var iTOP = 20;
 var iBOTTOM = 20;
 var iLEFT = 20;
 var iRIGHT = 20;
 var pMargin = [iTOP, iBOTTOM, iLEFT, iRIGHT];

 var aFont={type:"Times New Roman",size:15,underline:true};

 a.print({font:aFont,margin:pMargin},"This is a print test \nOn to nex
t
 Line");

 }
 catch(err) {
 System.TraceOutput(err);
 }
 }
printTest();
```

Usage Details

- Wsprint APIs
- Wsprint Functions
- Wsprint JavaScript Functions

- Printing a formatted receipt
- Printing tables

Unique solution ID: #1103
Author: Punil Shah
Last update: 2019-05-13 13:17