WS Basics
4.01 WS Special Scripts

WS special script files are entirely different from the script files associated with the SAP dynpros. These include the following.

Logon Script

The elogon script runs during the login process to SAP, as the name suggests. It contains anything that must be loaded or run at logon.

Session Script

The session script is very similar to the logon script and runs during session startup, but here the script runs each time you launch a new session. Anything that is specific to a session can be placed in this file.

Function Script

You can place any function that you wish to access in all transactions in the 'functionsSystem.sjs' script file. This special script file will help you in code optimization resulting in script quality.

Each of these script files is explained in detail in the following sections.

Logon Script (elogon.sjs)

You can include any functions or commands that need to be loaded during the logon process. This file is not required for a WS implementation, but it will be helpful when you need the file to be loaded or called at logon time. The logon script will be used in all your SAP transactions and is not restricted to a single session. An example logon script file is shown below.

elogon.sjs

```
load('wsoffice');
load('functionsSystem.sjs');
function getFilesToUpload(szPrompt)
{
    if(szPrompt==void 0) szPrompt = 'Select File';
    var dialog = new ActiveXObject('MsComDlg.CommonDialog');
    dialog.Filter='All Files(*.*)|*.*';
    dialog.MaxFileSize=32767;
    //dialog.AllowMultiSelect = true;
    dialog.DialogTitle=szPrompt;
    dialog.Flags=0x200|0x80000|0x800|0x4|0x200000
    dialog.ShowOpen();
    //var ret = dialog.FileTitle;
    var ret = dialog.FileName;
```

```
    dialog = void 0;
    return ret;
}
```

In this script file, You can see a set of libraries for the WSOffice extension, a set of functions housed in an external script file are called using load command and have a function that can access specific files for uploading. This particular logon script is from an Offline implementation, but you can use this script file in any interface.

The logon script is typically named as 'elogon.sjs'. This is the same regardless of the target language where e indicates English. The file will not be read, if you name the logon script anything other than 'elogon.sjs'.

Following are the commonly used commands in the logon script file.

'load' commands

'load' commands are used to load external files or libraries for use by the WS engine. Please see the load section for more information.

'onscreen' commands

'onscreen' commands specify a particular screen where an operation will occur. Please see the onscreen section for more information.

'set' commands

'set' commands are used to add value to a given object in SAP. Please see the set section for more information.

Note: You may also use other commands in logon scripts. However, remember that anything in a logon script will run at logon.

Session Script (esession.sjs)

The session script file (for English this is named 'esession.sjs') is used to contain any functions or commands that need to be loaded when you launch a new session in SAP. Like the logon script file, this file is not required for a WS implementation, but it is good practice to use it if you have elements that need to be loaded or called when a new session starts. Anything contained in the session script file will be used universally in your scripts for that particular session.

In the session script file, you can add any functions or commands that need to be loaded when you launch a new session in SAP. Like the logon script file, this file is not required for a WS implementation, but this session script file will be used in all your SAP transactions for that particular session.

Session scripts are named by prefixing a one-digit language code to the word 'session'. So for English, a session file would be named as 'esession.sjs', while for Japanese, the file would be named 'jsession.sjs', a French session script would be 'fsession.sjs' and so forth.

Like the 'elogon.sjs' file, there is no specific set of elements or commands that need to be included. The script in the file depends on the needs of a particular implementation. An example of esession.sjs file is shown below.

```
String.prototype.trim = function() { return this.replace(/^\s+|\s+$/g,
 "");} //Creates a string prototype function

currentUser = _user.toLowerCase().trim();
println('########## current user: ' + currentUser);

switch(currentUser) {


 case "maki":
  {
   currentDirectory = "d1=C:\\GuiXT\\WSScripts"  //Real Demo
   break;
  }

 case "demo1":
  {
   currentDirectory = "d1=C:\\GuiXT\\Demo3_Scripts" //Test1
   break;
  }

 case "demo2":
  {
   currentDirectory = "d1=C:\\GuiXT\\Scripts"   //Working folder
   break;
  }

 case "jasupport1":
  {
   currentDirectory = "d1=C:\\GuiXT\\Demo_Scripts"  //Training
   break;
  }

 case "jasupport2":
  {
   currentDirectory = "d1=C:\\GuiXT\\Demo2_Scripts" //Excel demo
   break;
  }

 default:
```

```
  {
   currentDirectory = "d1=C:\\GuiXT\\WSScripts"
   break;
  }

}
```

In this session script, we have created a switch statement so that for each of several different users, the default script directory will be set to the correct directory for that particular user when the session starts.

Functions Script(functionsSystem.sjs)

The functions script file is used to contain any functions or commands that you wish to make available to all script files. This makes easier script management by placing the functions in a single location. Using single file containing all your functions makes the scripts more portable. As this script file is independent of the transaction-specific script files, this allows code reusability for the same processes as the functions are available to any SAP transaction. An example from a functionsSystem.sjs file is shown below.

```
/*
** Open Excel file
*/
function open_excel(filename)
{
    if(g_ExcelApp == void 0)
 g_ExcelApp = new ActiveXObject('Excel.Application');
    g_ExcelBook = g_ExcelApp.Workbooks.Open(filename);
    g_ExcelApp.Visible = true;
    g_ExcelApp.ScreenUpdating = true;
}

/*
** get first item level column number (needs to be populated in to a t
able)
*/
var firstItemColumn;
function get_first_item_column(activesheet){
 //find firstItemColumn
 var lastCol = 3;
 for (var nCol=2; nCol < lastCol; nCol++){
  if (activesheet.Cells(1,nCol).Value.indexOf(",") > -1){
   firstItemColumn = nCol;
```

```
   break;
  } else {
   lastCol++;
  }
 }
}


/*
**Check if Screen Control exists str = "F[Material]", str = '#[3,0]'
*/
function isControl(str) {
 return Reebok(str).isValid;
}
```

In this function script, we have entered several functions, thus making them more portable as previously explained and enabling you to call them from any script file, instead of replicating them in each relevant script. In this case, we are performing operations related to the Excel data migration, but you can use the function script for any functions that you wish to have available to any transactions or processes.

Note: You need to ensure that this functions script file is loaded in the logon script file as previously demonstrated, otherwise the functions may not be available for reuse.

Unique solution ID: #1040
Author: Punil Shah
Last update: 2019-08-26 09:12