**ColdFusion + Excel: Harness the Power to Create, Read and Publish Professional Documents**

ColdFusion has a reputation for simplifying everyday programming tasks in convenient, easy to use native functions. With technology continuing to evolve and different systems finding common integration points, it is critical to make good technological decisions with your applications.

Excel documents are a common need in the business world and programmatically being able to work with them, with easy to use libraries, and efficiency is a must for any organization.

Fortunately, ColdFusion continues to excel at making these integrations as simple as possible for developers and users alike. With a few functions and a little coding, you can accomplish many Excel driven tasks in no time at all.

In the following tutorials we will look at some different use cases and potential applications that can be accomplished with ColdFusion with respect to Excel documents. First, we will review a handful of functions and then illustrate some code examples. Both tag based and script will be covered in the tutorials.

There are over 40 Adobe ColdFusion functions that can be used to process and manage Excel Documents. To keep focus, we will review some of the more important ones for reading, writing, deleting, and manipulating Excel Files.

*ColdFusion offers the ability to use tags and functions to manipulate spreadsheets. The cfspreadsheet tag is best used when reading in a spreadsheet; the spreadsheet functions are best used when manipulating the spreadsheet. There is the SpreadsheetRead function available to you, but it does not return the data contained in the spreadsheet. For this section, we will use the cfspreadsheet tag to read in data, but will use the spreadsheet functions for creation and manipulation.*

When exporting data, sometimes the data you are outputting needs to be separated. When dealing with spreadsheets, this is done by creating separate sheets in the spreadsheet file. To create a new sheet in your spreadsheet, you would use the SpreadsheetCreateSheet function. This function allows you to pass in the spreadsheet object and, optionally, the name to use for the sheet. Once this sheet is created, you will need to switch to the sheet before entering data. To do this, call the SpreadsheetSetActiveSheet function. This function accepts the spreadsheet object and the sheet name of the sheet you wish to set as the active sheet. Once this call is made, all updates will now occur on that sheet. As this function requires you provide the name of the sheet, and not the sheet position, it is recommended that you always name your sheets.

### Writing a Spreadsheet

Once you have added all the data to your spreadsheet and made all the necessary style and formatting updates, it will be necessary to write the spreadsheet to the file system. To do this, call the SpreadsheetWrite function.The SpreadsheetWrite function accepts the spreadsheet object as well as the location of where the spreadsheet should be written. When specifying the path, remember to include the file name.

SpreadsheetWrite also has some optional arguments that may be useful. In addition to being able to specify if the function should overwrite a pre-existing file if there already is one, you can also specify a password. This is a very useful feature when dealing with sensitive information.

### Spreadsheet Example

Below is an example of all the spreadsheet functionality discussed in this section. In this example, we create a spreadsheet, add some data to it, format the data, and then repeat this for a second sheet. Once completed, we then write the file to the file system.

//Create Spreadsheet

spreadsheetObj = SpreadsheetNew('Names');

//Add Header Row

SpreadSheetAddRow(spreadsheetObj,'ID,Name');

//Add Data

SpreadSheetAddRow(spreadsheetObj,'1,Simon');

SpreadSheetAddRow(spreadsheetObj,'2,Carl');

//Format Header

SpreadsheetformatRow(spreadsheetobj,{bold=true,alignment='center'},1);

//Add Sheet

SpreadSheetcreateSheet(spreadsheetobj,'Towns');

//Switch to Names Sheet

SpreadsheetSetActiveSheet(spreadsheetobj,'Towns');

//Add Header Row

SpreadSheetAddRow(spreadsheetObj,'ID,Town');

//Add Data

SpreadSheetAddRow(spreadsheetObj,'1,Detroit');

SpreadSheetAddRow(spreadsheetObj,'2,Sheffield');

//Format Header

SpreadsheetformatRow(spreadsheetobj,{bold=true,alignment='center'},1);

//Write File

Spreadsheetwrite(spreadsheetobj,expandpath('myData.xls'),true);

**ColdFusion Excel Functions in Focus**

# **Function Name:** IsSpreadsheetFile - Returns a value that determines if the input is a spreadsheet file.

# **Purpose:** Easily determine if the file you are prepping for management is a valid spreadsheet file.

**Syntax:**

**Script Based**

var fileBoolean = IsSpreadSheetFile(name of the file);

**Tag Based**

<cfset fileBoolean = IsSpreadSheetFile(name of the file)>

1. **Function Name:** IsSpreadsheetObject - Returns a value that determines if the input is a spreadsheet object.

**Purpose:** Easily determine if the file you are prepping for management is a valid spreadsheet object.

**Syntax:**

**Script Based**

var fileBoolean = IsSpreadSheetObject(name of the file);

**Tag Based**

<cfset fileBoolean = IsSpreadSheetObject (name of the file)>

1. **Function Name:** SpreadsheetInfo – Gets the properties of an excel spreadsheet object.

**Purpose:** A convenient way to return the properties of an excel document. The list is robust with the following properties: [

AUTHOR, CATEGORY, COMMENTS, CREATIONDATE, LASTEDITED, LASTAUTHOR, LASTSAVED, KEYWORDS, MANAGER, COMPANY, SUBJECT, TITLE, SHEETS, SHEETNAMES, SPREADSHEETTYPES]

**Syntax:**

**Script Based**

var fileInformation = spreadSheetInfo(spreadSheetObj);

var author = fileInformation.author;

**Tag Based**

<cfset fileInformation = spreadSheetInfo(spreadSheetObj)>

<cfset author = fileInformation.author>

1. **Function Name:** SpreadsheetNew - Creates a ColdFusion Excel spreadsheet object, which represents a single sheet of an Excel document.

**Purpose:** Easy to use function to generate an excel object for manipulation.

**Syntax:**

**Script Based**

var fileObj = spreadSheetNew([sheet name], [xml format]);

**Tag Based**

<cfset fileObj = spreadSheetNew ([sheet name], [xml format])>

**Note:** You can specify either "true" or "yes" to create a .xlsx file in the xml format argument.

1. **Function Name:** SpreadsheetRead - Reads a sheet from a spreadsheet file and stores it in a ColdFusion spreadsheet object.

**Purpose:**

**Syntax:**

**Script Based**

var spreadsheet = spreadSheetRead(fileName [, sheetName|sheet])

**Tag Based**

<cfset spreadsheet = spreadSheetRead(fileName [, sheetName|sheet])>

1. **Function Name:** SpreadsheetWrite - Writes single sheet to a new XLS file from a ColdFusion spreadsheet object.

**Purpose:**

**Syntax:**

**Script Based**

var spreadsheet = SpreadsheetWrite(fileName [, sheetName|sheet])

**Tag Based**

<cfset spreadsheet = SpreadsheetWrite(fileName [, sheetName|sheet])>

**ColdFusion Excel Examples**

For this section we will cover several examples of how to use ColdFusion Functions when working with spreadsheets. Both tag based and script will be covered, and tags snippets can be copied for ease of use. Source code files will be available as well.

**Create a New Spreadsheet Object and Write it to Disk**

**Important Note:** *If you are using ColdFusion 2021 make sure you have installed the Spreadsheet package. You can install the package through the CLI package manager or the ColdFusion Administrator GUI. If installing through the CLI package manager go to C:/ColdFusion2021/cfusion/bin/cfpm.bat and run the command : install Spreadsheet*

**

*Enter: install Spreadsheet*

<cfscript>

//create an excel spreadsheet file and provide a name as the single argument

//The first argument is the sheet name and the second is the format ( true = xlsx format)

students = SpreadsheetNew("Students", true);

//Set the headers for the xlsx file

SpreadsheetAddRow(students, "ID,First Name,Last Name,Year,GPA");

//Populate the sheet with data

SpreadsheetAddRow(students, "1,Gregory,Giroux,Senior,2.0");

SpreadsheetAddRow(students, "2,Mary,Flores,Junior,3.5");

SpreadsheetAddRow(students, "3,Donna,Williams,Senior,1.5");

SpreadsheetAddRow(students, "4,Joseph,Greene,Senior,4.0");

SpreadsheetAddRow(students, "5,Julia,Marin,Senior,4.0");

//Set the header for the file

//You can pass in an empty string in the places you do not require a header

//Example of all options for SpreadsheetSetHeader(students,"left header","center header","right header");

//Center will be selected in this instance

SpreadsheetSetHeader(students," ","Center header"," ");

//Save the file to disk

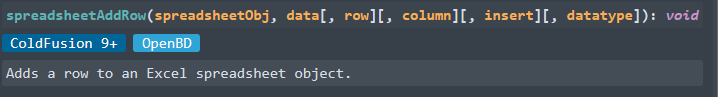
//Use the file path as the same location for your .cfm file

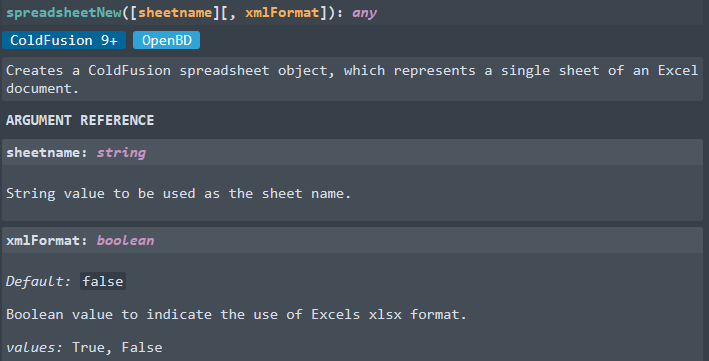
studentsFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "students.xlsx";

//Write the file to disk

SpreadsheetWrite(students,studentsFilePath,true);

</cfscript>

****

****

<cfscript>   
///We need an absolute path, so get the current directory path.   
theFile=GetDirectoryFromPath(GetCurrentTemplatePath()) & "newSpreadsheet.xls";   
//Create a new Excel spreadsheet object.   
theSheet = SpreadsheetNew("Expenses");   
//Set the value a cell.   
SpreadsheetSetCellValue(theSheet,"365",1,4);   
</cfscript>   
  
<!--- Write the spreadsheet to a file, replacing any existing file. --->   
<cfspreadsheet action="write" filename="#theFile#" name="theSheet" overwrite=true>

We will review the follow cases

1. Learn how to Read an Excel File

2. Read an Excel File as a Query

3. Write an Excel Spreadsheet

4. Convert a Form Into a Spreadsheet