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

Create, update, and analyze Excel Spreadsheets

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 critical to most organizations, and programmatically being able to manipulate them, with easy-to-use libraries, and efficiency, is a must for any company.

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 samples. 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.

This document isn’t just for reading, it’s for trying too. Watch for the Try it text in red throughout this document so you can learn by doing. Both Script and Tags will be covered!

Contents

[ColdFusion Functions in Focus 3](#_Toc67081505)

[IsSpreadsheetFile 3](#_Toc67081506)

[IsSpreadsheetObject 3](#_Toc67081507)

[SpreadsheetInfo 4](#_Toc67081508)

[SpreadsheetNew 4](#_Toc67081509)

[SpreadsheetRead 5](#_Toc67081510)

[ColdFusion Excel Examples 6](#_Toc67081511)

[Create a new spreadsheet and write it to disk 6](#_Toc67081512)

[Read an existing spreadsheet in ColdFusion code 12](#_Toc67081513)

[Add a sheet to an existing Excel File 15](#_Toc67081514)

[Retrieve a CSV file and convert to a .xlsx format 19](#_Toc67081515)

[Explore more features 22](#_Toc67081516)

[SpreadsheetInfo 23](#_Toc67081517)

[SpreadsheetGetColumnCount 24](#_Toc67081518)

[Closing thoughts 25](#_Toc67081519)

# ColdFusion Functions in Focus

## IsSpreadsheetFile

1. **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)>

## IsSpreadsheetObject

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)>

## SpreadsheetInfo

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>

## SpreadsheetNew

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.

## SpreadsheetRead

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

**Purpose:** Toread a spreadsheet file into a variable for manipulation.

**Syntax:**

**Script Based**

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

**Tag Based**

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

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

**Purpose:** To write multiple sheets to a single file and update an existing file. You can also read all sheets in the file, modify one or more sheets, and rewrite the entire file.

**Syntax:**

**Script Based**

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

**Tag Based**

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

1. **Function Name:** SpreadsheetAddRows - Adds multiple rows from a query to an Excel spreadsheet object

**Purpose:** To provide an easy way to write multiple rows without having to construct loops and calling multiple functions to add multiple sets of data.

**Syntax:**

**Script Based**

SpreadsheetAddrows(spreadsheetObj, data [, row, column, insert, datatype, includeColumnNames])

**Tag Based**

SpreadsheetAddrows(spreadsheetObj, data [, row, column, insert, datatype, includeColumnNames])

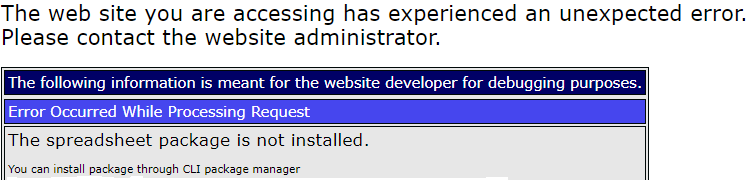
# ColdFusion Excel Examples

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

## Create a new spreadsheet and write it to disk

Try it

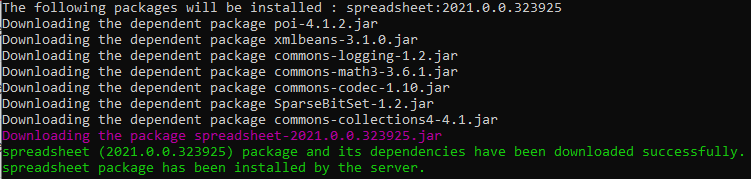
**Important Note:** *If you are using ColdFusion 2021 make sure you have installed the Spreadsheet package. Without the package you will receive the following error.*

**

*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*

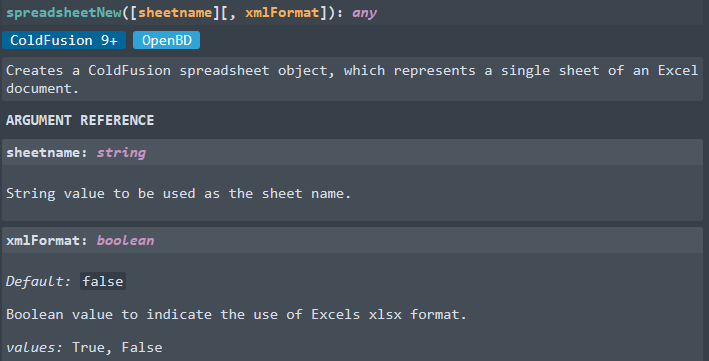
C:\>C:/ColdFusion2021/cfsuion/bin/cfpm.bat  
cfpm>install Spreadsheet



**Step 1:** Create an empty Spreadsheet Object and call it students. We will set the second argument to true so the format of xlsx is generated.

<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);  
</cfscript>

Below you will find the SpreadsheetNew function and its signature for further details.

****

**Step 2:** Set the headers to the Students object. We will use ID, First Name, Last Name, Year and GPA by using the SpreadsheetAddRow function.

<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");  
</cfscript>

**Step 3:** Add data to the Students Excel Object using the SpreadsheetAddRow. Maintain the positions of data by matching the headers.

<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");  
</cfscript>

**Step 4:** Use the SpreadsheetSetHeader to control the positioning of the headers. In this example, we will use “Center Header”.

<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>

**Step 5:** Now that we have populated the spreadsheet object, let us write it to disk for future manipulations. Use the GetDirectoryFromPath and GetCurrentTemplatePath functions to programmatically create the path based on the location of your .cfm file that contains your spreadsheet code. After you obtain the path use the SpreadsheetWrite function to write the file to disk. The second argument of true will overwrite an existing file with the same name.

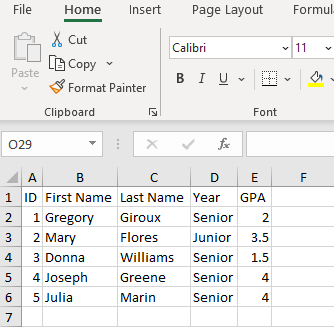
<cfscript>  
*//create an excel spreadsheet file and provide a name as the single argument  
//The first argument is the sheet name and the second is the format ( true = xlsx format)* students = SpreadsheetNew("Students", true);  
*//Set the headers for the xlsx file* SpreadsheetAddRow(students, "ID,First 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>

**Step 6:** Save your students.cfm file and the invoke it from a browser of your choice. This will generate a spreadsheet object. You should now have a file named students.cfm in your local directory. Open the file to confirm the data.

*File Name in local directory*



*File opened with data*



## Read an existing spreadsheet in ColdFusion code

Try it

In the previous sample we looked in depth at the process for creating a new spreadsheet. However, there will be cases where you want to read an excel file, make changes, and save the updated file to disk. In this section we will explore opening/reading a file in code, manipulating the file with data, and then saving that file back to disk. Let’s begin.

When we created the Students.xlsx file we saved the file to the same directory where we executed our .cfm file. To keep things easy, we will use that same file and programmatically set it to a variable.

**Step 1:** Create a new .cfm file called ExcelReader.cfm. Save the file to the same directory as the Students.cfm file.

**Step 2:** In code, use the GetDirectoryFromPath function and the GetCurrentTemplatePath function to dynamically obtain the path of the Students.xlsx file from the previous tutorial.

<cfscript>  
 *//We need an absolute path, so get the current directory path.* studentsFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "Students.xlsx";  
</cfscript>

**Step 3:** Set the file object to studenstFilePath and pass that argument in to the SpreadsheetRead function. We will set that variable to studentsFile.

<cfscript>  
 *//We need an absolute path, so get the current directory path.* studentsFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "Students.xlsx";  
 *//Read an existing Excel spreadsheet into a ColdFusion Excel Object* studentsFile = SpreadSheetRead(studentsFilePath);  
</cfscript>

**Step 4:** You will now have the studentsFile object ready to manipulate. You can debug the code by printing out a dump to review the data. Use the writeDump function to view the data.

<cfscript>  
 *//We need an absolute path, so get the current directory path.* studentsFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "Students.xlsx";  
 *//Read an existing Excel spreadsheet into a ColdFusion Excel Object* studentsFile = SpreadSheetRead(studentsFilePath);  
 *//dump the data from the stream* writeDump(studentsFile);  
</cfscript>

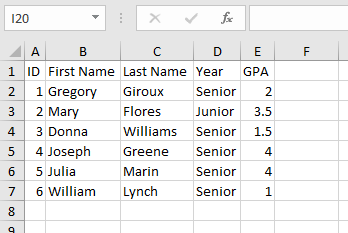
**Step 5:** Let’s say a student has been added to the roster. We need to manipulate the excel document to add the student. We will use the SpreadsheetAddRow to perform this task.

<cfscript>  
 *//We need an absolute path, so get the current directory path.* studentsFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "Students.xlsx";  
 *//Read an existing Excel spreadsheet into a ColdFusion Excel Object* studentsFile = SpreadSheetRead(studentsFilePath);  
 *//Add a new student to the ColdFusion Excel Object* SpreadsheetAddRow(studentsFile, "6,William,Lynch,Senior,1.0");  
</cfscript>

**Step 6:** Now that we have added the additional student, let us save the file to disk. We will use the same function as before. Earlier, we set the file path. We can re-use that variable when writing the file. As you can see in the example below, we will provide the file name, the path, and the overwrite flag set to true.

<cfscript>  
 *//We need an absolute path, so get the current directory path.* studentsFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "Students.xlsx";  
 *//Read an existing Excel spreadsheet into a ColdFusion Excel Object* studentsFile = SpreadSheetRead(studentsFilePath);  
 *//Add a new student to the ColdFusion Excel Object* SpreadsheetAddRow(studentsFile, "6,William,Lynch,Senior,1.0");  
 *//Write the file to disk  
 //Reuse the path from earlier for the write location/destination. Use the true argument to overwrite the existing file* SpreadsheetWrite(studentsFile,studentsFilePath,true);  
</cfscript>

**Step 7:** Open the file Students.xlsx and verify that the data has been correctly added.



## Add a sheet to an existing Excel File

Try it

In our previous examples, we created an excel document with a single sheet. There are cases where you may want multiple sheets. ColdFusion makes this a simple task. We will take our existing sheet and add a new sheet called “Subjects”. We will name the first sheet “Summary” to separate the responsibilities.

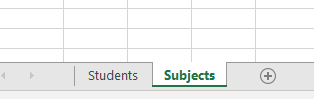
**Step 1:** First, we need to read the spreadsheet into a ColdFusion Object. We will stick with the same naming conventions to build upon our progress.

<cfscript>  
 *//We need an absolute path, so get the current directory path.* studentsFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "Students.xlsx";  
 *//Read an existing Excel spreadsheet into a ColdFusion Excel Object* studentsFile = SpreadSheetRead(studentsFilePath);  
</cfscript>

**Step 2:** The SpreadSheetCreateSheet is a convenient function to add a sheet to an existing workbook. Since we have created a ColdFusion Excel object, we can easily execute this function to prepare a second sheet. We will reuse the sheet from the previous exercise.

<cfscript>  
 *//We need an absolute path, so get the current directory path.* studentsFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "Students.xlsx";  
 *//Read an existing Excel spreadsheet into a ColdFusion Excel Object* studentsFile = SpreadSheetRead(studentsFilePath);  
 *//Use the SpreadSheetCreateSheet to add a sheet called "Subjects"* SpreadSheetCreateSheet(studentsFile, "Subjects");  
 *//Write the file to disk  
 //Reuse the path from earlier for the write location/destination. Use the true argument to overwrite the existing file* SpreadsheetWrite(studentsFile,studentsFilePath,true);  
</cfscript>

**Step 3:** Verify that the sheet was created properly in the Students.xlsx file. You should now see the book called Subjects.



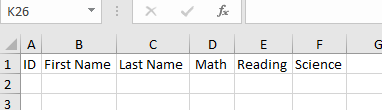
**Step 4:**  Let us create the headers for the new sheet. We will use the SpreadsheetAddRow function and declare the values for the headers. In this example, ID, First Name, Last Name, Math, Reading, and Science will be the column headers.

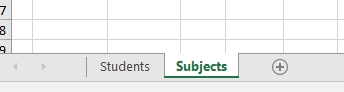
<cfscript>  
 *//We need an absolute path, so get the current directory path.* studentsFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "Students.xlsx";  
 *//Read an existing Excel spreadsheet into a ColdFusion Excel Object* studentsFile = SpreadSheetRead(studentsFilePath);  
 *//Set the active sheet to Subjects* SpreadsheetSetActiveSheet(studentsFile,"Subjects");  
 *//Set the headers for the second sheet* SpreadsheetAddRow(studentsFile, "ID,First Name,Last Name, Math,Reading,Science");  
 *//Write the file to disk  
 //Reuse the path from earlier for the write location/destination. Use the true argument to overwrite the existing file* SpreadsheetWrite(studentsFile,studentsFilePath,true);  
</cfscript>

**Step 4:** Now to populate the new sheet with some data and save it to the disk. We will again utilize the SpreadsheetAddRow function to begin adding data to the sheet.

<cfscript>  
 *//We need an absolute path, so get the current directory path.* studentsFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "Students.xlsx";  
 *//Read an existing Excel spreadsheet into a ColdFusion Excel Object* studentsFile = SpreadSheetRead(studentsFilePath);  
 spreadsheetSetActiveSheet(studentsFile, "Subjects");  
 *//Add headers as a row of data to the Subjects book/sheet using the Add Row function* SpreadsheetAddRow(studentsFile, "ID,First Name,Last Name, Math,Reading,Science");  
 *//Write the file to disk  
 //Reuse the path from earlier for the write location/destination. Use the true argument to overwrite the existing file* SpreadsheetWrite(studentsFile,studentsFilePath,true);  
</cfscript>

**Step 6:** Verify that the file has been correctly updated and committed to disk. Verify that the file is in the correct destination and it contains the correct data based on the exercise above.

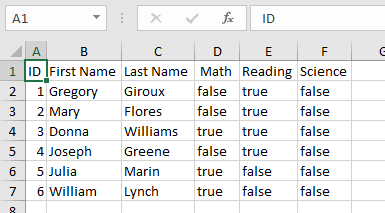




**Step 7:** Add some data to the active “Subjects” sheet.

<cfscript>  
 *//We need an absolute path, so get the current directory path.* studentsFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "Students.xlsx";  
 *//Read an existing Excel spreadsheet into a ColdFusion Excel Object* studentsFile = SpreadSheetRead(studentsFilePath);  
 spreadsheetSetActiveSheet(studentsFile, "Subjects");  
 *//Add headers as a row of data to the Subjects book/sheet using the Add Row function* SpreadsheetAddRow(studentsFile, "1,Gregory,Giroux,false,true,false");  
 SpreadsheetAddRow(studentsFile, "2,Mary,Flores,false,true,false");  
 SpreadsheetAddRow(studentsFile, "3,Donna,Williams,true,true,false");  
 SpreadsheetAddRow(studentsFile, "4,Joseph,Greene,false,true,false");  
 SpreadsheetAddRow(studentsFile, "5,Julia,Marin,true,false,false");  
 SpreadsheetAddRow(studentsFile, "6,William,Lynch,true,false,false");  
 *//Write the file to disk  
 //Reuse the path from earlier for the write location/destination. Use the true argument to overwrite the existing file* SpreadsheetWrite(studentsFile,studentsFilePath,true);  
</cfscript>

**Step 8:** Verify the sheet.



## Retrieve a CSV file and convert to a .xlsx format

Try it

In this tutorial, we will be reading the contents of a CSV file, manipulating the data, and saving it to disk. Create a file called csvRead.cfm

You will be using a cfhttp call to retrieve the data from GitHub which will have a sample CSV file ready for consumption.

**Step 1:** Use the cfhttp function to retrieve the contents of the csv file from GitHub. The permalink is:

**https://raw.githubusercontent.com/bsappey/ColdFusionExcelDemos/main/excelDemo.csv**

<cfscript>  
 *//retreieve csv data format from github* cfhttp( name="csvData", url="https://raw.githubusercontent.com/bsappey/ColdFusionExcelDemos/main/excelDemo.csv", firstrowasheaders="true" ,method="GET");  
 writeDump(csvData);abort;  
</cfscript>

**Step 2:** Get the current working directory based on the csvRead.cfm file location and set it to a variable. Create a new ColdFusion Excel object and set it to a variable called csvSheet. The sample .XLSX file will be called CSVExample.xlsx

<cfscript>  
 *//retrieve csv data format from github* cfhttp( name="csvData", url="https://raw.githubusercontent.com/bsappey/ColdFusionExcelDemos/main/excelDemo.csv", firstrowasheaders="true" ,method="GET");  
 *//We need an absolute path, so get the current directory path and name for the write to disk* csvFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "CSVExample.xlsx";  
 *//Create an empty ColdFusion spreadsheet object and call it CSVExample.xlsx* csvSheet = SpreadsheetNew("CSVExample.xlsx", true);  
</cfscript>

**Step 3:** Now you will add all the csv data from the REST call to the ColdFusion Excel Object by using the function spreadSheetAddrows. This will allow bulk data to be passed in through one function.

<cfscript>  
 *//retrieve csv data format from github* cfhttp( name="csvData", url="https://raw.githubusercontent.com/bsappey/ColdFusionExcelDemos/main/excelDemo.csv", firstrowasheaders="true" ,method="GET");  
 *//We need an absolute path, so get the current directory path and name for the write to disk* csvFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "CSVExample.xlsx";  
 *//Create an empty ColdFusion spreadsheet object and call it CSVExample.xlsx* csvSheet = SpreadsheetNew("CSVExample.xlsx", true);  
 *//Add the csv data from the git hub api request and write it to the new spreadsheet object* SpreadSheetAddRows( csvSheet, csvData );  
</cfscript>

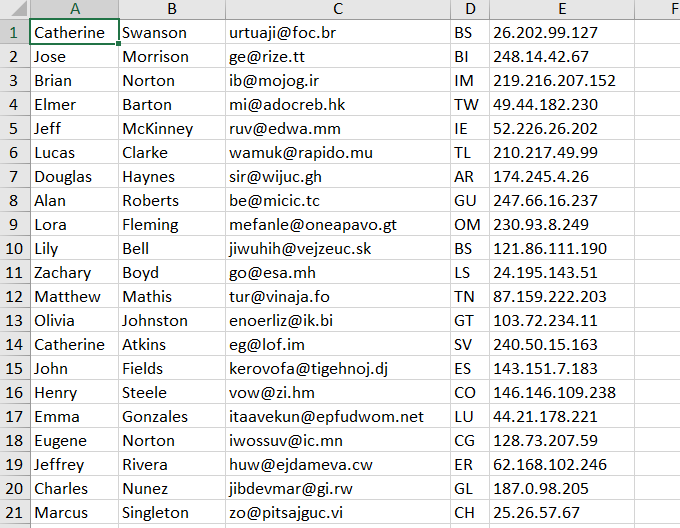
**Step 4:** Write the file to disk. You should have all the data now published in the excel file called CSVExample.xlsx

<cfscript>  
 *//retrieve csv data format from github* cfhttp( name="csvData", url="https://raw.githubusercontent.com/bsappey/ColdFusionExcelDemos/main/excelDemo.csv", firstrowasheaders="true" ,method="GET");  
 *//We need an absolute path, so get the current directory path and name for the write to disk* csvFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "CSVExample.xlsx";  
 *//Create an empty ColdFusion spreadsheet object and call it CSVExample.xlsx* csvSheet = SpreadsheetNew("CSVExample.xlsx", true);  
 *//Add the csv data from the git hub api request and write it to the new spreadsheet object* SpreadSheetAddRows( csvSheet, csvData );  
 *//Write the file to disk* SpreadsheetWrite(csvSheet,csvFilePath,true);  
</cfscript>

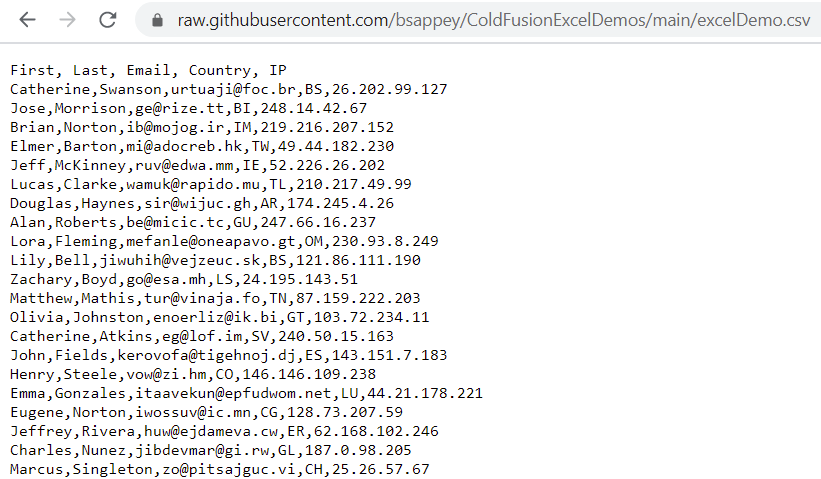
**Step 5:** In the root directory which contains the csvRead.cfm file, you should now see an excel file called CSVExample.xlsx



**Step 6:** Open the file an verify that the data is correct, and the file is properly formatted



**Step 7:** You can cross reference the accuracy of the file by invoking the GitHub Raw Link for the CSV data.



# Explore more features

We have explored a few ways to create, update and write spreadsheets using ColdFusion. There many use cases where Excel Spreadsheets need to be quickly generated and manipulated for everyday practicality in a professional, personal, or educational setting.

ColdFusion also offers several functions to quickly gain insight into a spreadsheet or simply inspect data programmatically without the need for manual intervention. The ability to automate and further obtain details without the need of manual file sends, opens, and reads will certainly cut down on the amount of labor needed in day-to-day activities.

Let’s review a few other functions that can be useful when working with spreadsheets and ColdFusion.

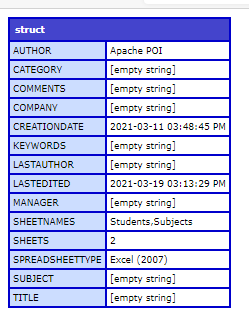
## SpreadsheetInfo

Try it

Obtaining properties about a spreadsheet can be easily accomplished with the use of ColdFusion. By simply passing an excel object into the SpreadsheetInfo function, you can retrieve several properties about a file without having to manually read and open to view this information. Let’s take a quick review of what can be accomplished with this function.

**Step 1:** Using the same file we created earlier Students.xlsx. Let’s open the file and persist it to a ColdFusion Excel Object.

<cfscript>  
*//We need an absolute path, so get the current directory path.* studentsFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "Students.xlsx";  
*//Read an existing Excel spreadsheet into a ColdFusion Excel Object* studentsFile = SpreadSheetRead(studentsFilePath);  
*//use the SpreadsheetGetColumnCount to get a column count* data = spreadsheetInfo(studentsFile);  
 writeDump(data);  
</cfscript>



## SpreadsheetGetColumnCount

Try it

**Step 1:** Create a file called Utilities.cfm and use the following code. This should return the count of 6.

<cfscript>  
 *//We need an absolute path, so get the current directory path.* studentsFilePath=GetDirectoryFromPath(GetCurrentTemplatePath()) & "Students.xlsx";  
 *//Read an existing Excel spreadsheet into a ColdFusion Excel Object* studentsFile = SpreadSheetRead(studentsFilePath);  
 *//use the SpreadsheetGetColumnCount to get a column count* numberOfColumns = spreadsheetGetColumnCount(studentsFile);  
 writeDump(numberOfColumns);  
</cfscript>

# Closing thoughts

We have only scratched the surface in the above tutorials. ColdFusion simplifies the process of working with Excel. Without the need for complicated libraries and dependencies, you can be up and running with a few functions and begin manipulating Excel based objects. Whether you are creating excel documents from scratch, from a known data source, or simply modifying Excel objects, with a few lines of code, your Excel journey can begin.

ColdFusion offers such a high level of abstraction when working with these types of documents. By combining the simplicity of the package manager and ColdFusion built-in functions, the learning curve is small. You will spend more time coding and executing tasks on hand rather than researching dependencies and performing expensive installs and updates. Within minutes you can be off the launch pad and working away on your new project or assignment.

Even in the age of big data, Excel is still very relevant. Think of it this way, it is a fantastic collaborative tool and a central framework for many people when working with datasets in a day-to-day environment. With that fact in hand, think of the benefits of programmatically being able to manipulate these collaborative data sets without any manual involvement. ColdFusion certainly can make your life easier.