# Excel Export Helper

## Overview

Excel Export Helper (EEH) is a Java library that is used to easily output application data to a Microsoft Excel spreadsheet file. The EEH uses the Apache POI library to create the Excel files, but the EEH abstracts out the implementation detail of Apache POI, so a developer using the EEH only needs to use a very simple API to out put application data to Excel very quickly.

## Technical Information:

* The EEH was written in Java 8
* The EEH uses Apache POI 5.0.0

## Quick Start

Create an abject instance of the EEH class ExcelExportHelper, providing it a file to be populated with data. On the object instance of this class, call the method createSheet to create a sheet object, and provide the sheet with a name for the sheet. The sheet object has an internal list of lists of strings that can be populated that will result as the cell data when the file is written. More sheets can be created as needed. After the cell data has been completely entered, call the writeFile method on the ExcelExportHelper object to write the data to a new Excel spreadsheet file.

## To create the ExcelExportHelper:

To create the main ExcelExportHelper object, a File object or full String file path must be provided. There are two constructors for the ExcelExportHelper class, one that takes a File object, and one that takes a String as a parameter. The name of the file must not be longer than 31 characters. If the given file name exceeds the maximum character length, the file name will be truncated at the end of the name. The name of the file needs to end with “.xlsx”, if the given filename does not end with this suffix, the suffix will automatically be appended to the end of the filename. The file path given must be at a writable location. If the location set for where the file is to be written is not writable, an IllegalArgumentException will be thrown. The filename must be at least one character long. If the filename is empty, an IllegalArgumentException will be thrown.

## To create a sheet:

To create a sheet to populate with Excel cell data, call the createSheet method on the ExcelExportHelper object reference. The createSheet method takes a String value as the name for the new sheet. If the name already exists in the EEH instance, then the EEH will modify the name with an appended digit, (ie: Sheet, and Sheet1). The minimum length for a sheet name is one character, and the maximum number of sheets possible in an Excel file is 255.

## To populate a sheet:

The EEHSheet object reference that is returned contains an internal list of lists of Strings that hold the Excel sheet cell data. A row of data in the final Excel sheet is represented by one of the lists of Strings in the EEHSheet object reference. To add a row of sheet data, create an ArrayList of Strings, and then add it to the sheet in the typical fascion of adding an object to a list.

EEHSheet sheet = eeh.createSheet("Sheet A");

ArrayList<String> data = new ArrayList<>();

data.add("One");

data.add("Two");

data.add("Three");

sheet.getData().add(data);

## To create a header row for a sheet:

The EEH features the ability to create a header row of bold text for an Excel sheet. The contents of the header row are stored as an ArrayList of Strings. To set a header row for the sheet, pass a string to the add method of the getHeaders() method on the EEHSheet object reference. The header row is set in the top most row in the excel sheet. If the list of headers in a sheet is empty, the header row is not set. Only one header row can be set for an Excel sheet.

## To write the data to the Excel file:

After the EEH instance has been populated with sheet data, call the writeWorkBook() method to trigger the EEH to write the data to a new Excel file. If the writeWorkBook() method is called before any sheets are created, then an exception will be thrown.

## To load the EEH library:

For the Excel Export Helper library to work, the Apache POI 5.0.0 library must be loaded in your project.

Include the follow dependency into your maven pom.xml file:

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>5.0.0</version>

</dependency>

Or download the jar files from the following website and load them into your class path:

https://jar-download.com/artifacts/org.apache.poi