Simple Excel

SimpleExcel is intended to be used to create quick and easy Excel downloads for reporting purposes. Any array of homogenous objects can be converted into an Excel spreadsheet.

# Column Types

*define*(**'SIMPLE\_EXCEL\_PROPERTY\_TYPE\_CALCULATED'**, 0);  
*define*(**'SIMPLE\_EXCEL\_PROPERTY\_TYPE\_AS\_GIVEN'**, 1);  
*define*(**'SIMPLE\_EXCEL\_PROPERTY\_TYPE\_DATE'**, 2);  
*define*(**'SIMPLE\_EXCEL\_PROPERTY\_TYPE\_DATETIME'**, 3);

## Calculated:

The value put into the Excel spreadsheet is whatever Excel interprets the source value to be. This is the default behavior and works well enough for most numbers and strings.

## As Given:

The value put into the Excel spreadsheet will be exactly what the source value is. This is useful when source data has leading zeros, or it could be a long account identifier and Excel would otherwise interpret it as a very large number and change it to scientific notation.

## Date:

The value will be put into Excel as only the date portion. For example, 2012-01-01. By default, Excel will add the time information to a date even if the time is not given in the source data. It will just be zeros.

## Datetime:

The value will be put into Excel with the date and time. Currently this is unhandled by Simple Excel since the default behavior is to put the Date and Time in Excel. This is more to document your data types and allow for special handling in the future.

# Configuring Routing

**if**($Request->**export**) {  
 **switch**(*strtolower*($Request->**export**)) {  
 **case 'xls'**:  
 **if**(*method\_exists*($PageModel, **'ExportToXLS'**)) {  
 $PageModel->ExportToXLS();  
 } **else** {  
 **exit**(**'ExportToXLS Not Implemented: '** . *get\_class*($PageModel));  
 }  
 **exit**;  
 **case 'json'**:  
 **if**(*method\_exists*($PageModel, **'ToJSON'**)) {  
 $PageModel->ToJSON();  
 } **else** {  
 **exit**(**'ToJSON Not Implemented'**);  
 }  
 **exit**;  
 }  
}

After GET or POST are processed, this block of code checks to see if the export http parameter has been set. If so, it checks the type requested and looks for the corresponding method on the PageModel class. If it doesn’t exist, the request is ignored.  
  
This allows you to create your page, output the report to HTML Tables and then have a simple link that adds “&export=xls” that the user can click to trigger the ExportToXLS method and present a downloaded file to the user.

Since the check for export request is after the GET / POST check, you can use a POST or GET action to trigger the logic to generate the report and push it to an Excel file.

# Exporting Reports with ExportToXLS:

**public function** ExportToXLS()  
{  
 $se = **new** SimpleExcel();  
 $se->**Columns** = [  
 **new** SimpleExcel\_Column(**'Order No'**,**'Order\_No'**, ***SIMPLE\_EXCEL\_PROPERTY\_TYPE\_AS\_GIVEN***),  
 **new** SimpleExcel\_Column(**'Order Line No'**,**'Order\_Line\_No'**),  
 **new** SimpleExcel\_Column(**'Description'**,**'Description'**),  
 **new** SimpleExcel\_Column(**'Amt Paid'**,**'AmtPaid'**),  
 **new** SimpleExcel\_Column(**'First Name'**,**'First\_Name'**),  
 **new** SimpleExcel\_Column(**'Last Name'**,**'Last\_Name'**),  
 **new** SimpleExcel\_Column(**'Date Paid'**,**'DatePaid'**, ***SIMPLE\_EXCEL\_PROPERTY\_TYPE\_DATE***),  
 **new** SimpleExcel\_Column(**'Ship Cust Id'**,**'ShipCustId'**, ***SIMPLE\_EXCEL\_PROPERTY\_TYPE\_AS\_GIVEN***),  
 **new** SimpleExcel\_Column(**'Bill Cust Id'**,**'BillCustId'**, ***SIMPLE\_EXCEL\_PROPERTY\_TYPE\_AS\_GIVEN***),  
 **new** SimpleExcel\_Column(**'Addr 1'**,**'Addr1'**),  
 **new** SimpleExcel\_Column(**'Addr 2'**,**'Addr2'**),  
 **new** SimpleExcel\_Column(**'Addr 3'**,**'Addr3'**),  
 **new** SimpleExcel\_Column(**'Addr 4'**,**'Addr4'**),  
 **new** SimpleExcel\_Column(**'City'**,**'City'**),  
 **new** SimpleExcel\_Column(**'St'**,**'St'**),  
 **new** SimpleExcel\_Column(**'Zip'**,**'Zip'**),  
 **new** SimpleExcel\_Column(**'Country'**,**'Country'**),  
 ];  
 $se->**Report** = $this->**Receipt**->**LineItems**;  
 $se->**Filename** = **'Receipt.'** . $this->**ReceiptNumber** . **'.xlsx'**;  
 $se->**Title** = **'Receipt '** . $this->**ReceiptNumber**;  
 SimpleExcel::*SingleSheet2007*($se);  
}

SimpleExcel\_Column takes at minimum 2 parameters: the title and the property. The order of the array determines the order of the columns in the resulting Excel file. An optional 3rd parameter is the Column Type.

Report is the array of homogenous objects that will be put into the Excel spreadsheet. Only one type of object can be used per worksheet. If a column references a non-existent property, the export will fail.

Filename is the name of the file that the spreadsheet will download as.

Title is the name of the worksheet in Excel. Note that there is a character limit of about 30 that is not handled by SimpleExcel. If your title is too long it will error out rather than quietly truncate it.

There are four output functions to create a Excel file from the SimpleExcel object.

## Single Sheet

SimpleExcel::SingleSheet2007($se)

SingleSheet2007 will output an XLSX file. Note that the filename you specify must have the correct extension given or Excel will complain that the file doesn’t match the extension.

SimpleExcel::SingleSheet ($se)

SingleSheet will output an Excel 2005 formatted file. The filename given must be XLS or Excel will yell at you.

## Multi-Sheet

SimpleExcel:: MultiSheet($filename, $reports)

SimpleExcel:: MultiSheet2007($filename, $reports)

The MultiSheet functions are named to indicate whether you want an Excel 2005 format file or newer version.

Rather than setting the Filename through the SimpleExcel object, it is passed into the function and $reports is an array of SimpleExcel objects.  
  
This is used when you have a more complex report that has multiple datasets associated with it.

# SimpleExcel\_Reader

This object is used to make it easy to load an Excel compatible file uploaded by a user or loaded through cron jobs.

SimpleExcel\_Reader::FromFilename(

$file, $process\_cells = **true**, $debug = **false**, $row\_limit = **null**

)

## File

File is the path to the file being read.

## Process Cells

The PHPExcel class can be very inefficient when it comes to more complex workbooks. The vlookup function is particularly slow. In instances where you don’t want PHPExcel to interpret cells and just give you the raw values, you can set this parameter to false. When process\_cells is false, equations will be returned rather than the calculated value of those equations.

One of issues with PHPExcel is that you can either process cells or not. There is no fine grain control. And Excel stores all dates in a special format so you must turn on Process Cells in order to read the date. You cannot tell it to process dates but not for example, the vlookup.

If possible, your best option is to use Excel to export your worksheet to CSV before using this function to import it if you have large data sets.

This function will assume that the first row with most columns set is the header row. It allows for up to 2 missing values in the header. Again, Excel workbooks need to be properly formatted for best results. It is impossible to account for all types of sloppy work. The header should be the first row on each worksheet and every column should have a header.

This function returns an associative array using the worksheet name as the key for each array of data. The first row of data is the header and all the actual data follows. This function does not convert the data to be associative. This is up to the end user so that even if header names are duplicated, it will import correctly.

## Debug

When Debug is true, various debug information will be output to the console so that you can verify that large files are being processed

## Row Limit

If you simply want schema information to build databases based on your Excel files, then you can set the row limit small so that you can quickly get through your files to get the header information you need to create tables to store the spreadsheets with.