

From Stored Procs to Self-Serve

Contents

Step 1: Create the stored procedure in SQL Server Management Studio	3
Step 2: Generate Code	4
Step 3: Set Up a Page	5
Step 4: Make the Page Accessible.....	7
Step 5: Build Out the Form	9
Step 6: Handle the Form	10
Step 7: Pass the Form Values to the Stored Procedure	11
Step 8: Defining the Properties of the Stored Procedure	12
Step 9: Render the Results to the User.....	14
Step 10: Allow the User to Export the Report to XLSX.....	20
Step 11: Tidy Up	24

Step 1: Create the stored procedure in SQL Server Management Studio

<http://poorsql.com/>

This is an excellent resource for formatting queries consistently for readability.

http://architectshack.com/PoorMansTSqlFormatter.ashx#Download_4

There is a WinForms App which is a simple app that doesn't need to be installed that can be run locally and is better at handling large, complex queries.

Step 2: Generate Code

In Persephone / scripts there is a bat file named regen_ismweb.bat

Other bat files are set to generate code for other databases including all the personify databases.

You can run these as many times as you need. Generated code will go directly into the httpdocs folder and will not overwrite files that should not be overwritten.

Step 3: Set Up a Page

The Stored Procedure we'll be making into a self-serve page is "Ismweb.dbo.GetStTaxOrders"

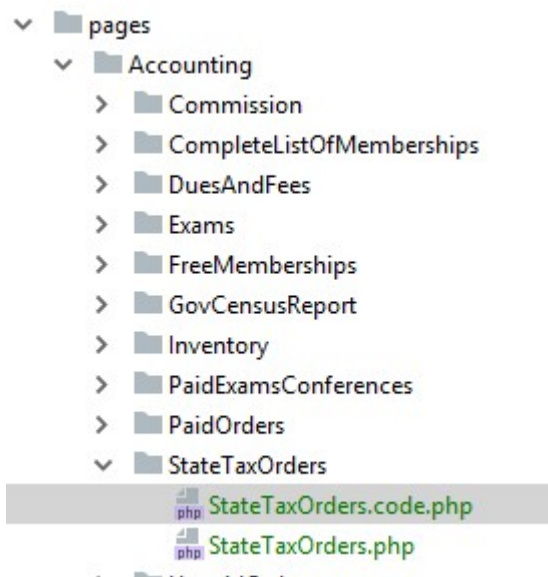
This goes under Accounting so we'll start by making a new folder in the pages / Accounting folder named

StateTaxOrders

We'll then create two files in the folder named

StateTaxOrders.code.php

StateTaxOrders.php



In StateTaxOrders.code.php create a class structure

```

1  <?php
2
3  /**
4   * Class StateTaxOrders
5   */
6  class StateTaxOrders extends BasePage
7  {
8      public static function DoInit()
9      {
10         self::$MasterPage = MASTERPAGE_USER_SIDEMENU;
11     }
12 }
13
14 $Web->PageMode = QUICKDRY_MODE_STATIC;
15

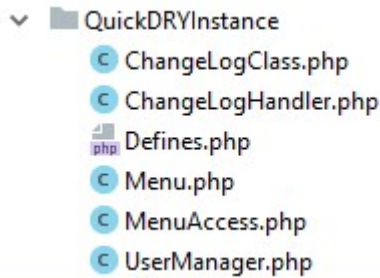
```

This is the minimum code needed for a page to load. DoInit() is called before permissions are checked. The masterpage is used to determine whether or not a page is secure. It is best not to do any business logic in this function. Checking for valid request parameters is fine.

The class name must match the file name exactly and the file name must match the folder name exactly. This is how the framework knows what the name of the class is when rendering the page.

BasePage makes the class compatible with the framework and gives you access to a number of things needed to render a page.

Step 4: Make the Page Accessible



In httpdocs / QuickDRYInstance there are two files we need to modify to make the page accessible to users. First, we need to add the file to Menu.php



Here we have added “State Tax Orders” to the Accounting menu item and pointed it at “/Accounting/StateTaxOrders”

Notice that we are matching the path structure under the pages folder. The order does not matter because the links will be alphabetized automatically.

Highlight and copy “/Accounting/StateTaxOrders”

Then open MenuAccess.php

Find the accounting section and paste in your link. Then add in the array of Role IDs that will have access to the page. In this case we are giving Administrators and Accounting access to the page.

```

'/Accounting/CompleteListOfMemberships' => [ROLE_ID_ADMIN, ROLE_ID_ACCOUNTING],
'/Accounting/FreeMemberships' => [ROLE_ID_ADMIN, ROLE_ID_ACCOUNTING],
'/Accounting/PaidExamsConferences' => [ROLE_ID_ADMIN, ROLE_ID_ACCOUNTING],
'/Accounting/Exams' => [ROLE_ID_ADMIN, ROLE_ID_ACCOUNTING],
'/Accounting/GovCensusReport' => [ROLE_ID_ADMIN, ROLE_ID_ACCOUNTING],
'/Accounting/UnpaidOrders' => [ROLE_ID_ADMIN, ROLE_ID_ACCOUNTING, ROLE_ID_SALES],
'/Accounting/PaidOrders' => [ROLE_ID_ADMIN, ROLE_ID_ACCOUNTING, ROLE_ID_SALES],
'/Accounting/Inventory' => [ROLE_ID_ADMIN, ROLE_ID_ACCOUNTING, ROLE_ID_SALES],
'/Accounting/Commission' => [ROLE_ID_ADMIN, ROLE_ID_ACCOUNTING],
'/Accounting/StateTaxOrders' => [ROLE_ID_ADMIN, ROLE_ID_ACCOUNTING],

```

Now, when you sign into reports.localhost.com, the page will show up in the accounting menu



Step 5: Build Out the Form

Persephone uses Bootstrap 3 and so panels are used to wrap content.

```
1 <div class="panel panel-default">
2   <div class="panel-heading">
3     <div class="panel-title">State Tax Orders</div>
4   </div>
5   <div class="panel-body">
6     <form method="get">
7       <label>Start Date: <input type="text" class="form-control date-picker" name="start_date" /></label>
8       <label>End Date: <input type="text" class="form-control date-picker" name="end_date" /></label>
9       <label>State: <?php echo FormClass::SelectItems(StatesClass::GetStates(), selected: null, id: 'state', class: 'form-control'); ?></label>
10      <input type="submit" class="btn btn-primary" />
11    </form>
12  </div>
13 </div>
14
```

There is a css class 'date-picker' that will turn a text entry box into a date selection box. There is a FormsClass that makes it easier to turn lists into selection options for the user.

Note that the "name" of the form elements must be valid PHP variables. No spaces or special characters allowed. Underscores are preferred to make variable names readable and lower case is preferred because the framework is case sensitive when it comes to request parameters.

Now that your initial form is created, put in some test values and hit submit.

Step 6: Handle the Form

```
1 <?php
2
3 /**
4  * Class StateTaxOrders
5  */
6 class StateTaxOrders extends BasePage
7 {
8     public static $StartDate;
9     public static $EndDate;
10    public static $State;
11
12    public static function DoInit()
13    {
14        self::$MasterPage = MASTERPAGE_USER_SIDEMENU;
15    }
16
17    public static function DoGet()
18    {
19        self::$State = self::$Request->state;
20        self::$StartDate = Dates::Datestamp(self::$Request->start_date, null: '');
21        self::$EndDate = Dates::Datestamp(self::$Request->end_date, null: '');
22    }
23 }
24
25 $Web->PageMode = QUICKDRY_MODE_STATIC;
26
27
```

The first thing to do is set up some static properties to hold the user's input. There is a Dates class which handles converting user input to a SQL compatible string and other formats. If the null value is left out, then it will default to today's date.

With those defined, we can go back to our form and set the default values in the form using the properties of the StateTaxOrders class. We do not want to use Request directly in the form because we want to sanitize user input.

```
1 <div class="panel panel-default">
2     <div class="panel-heading">
3         <div class="panel-title">State Tax Orders</div>
4     </div>
5     <div class="panel-body">
6         <form method="get">
7             <label>Start Date: <input type="text" class="form-control date-picker" name="start_date" value="<php echo Dates::StandardDate(StateTaxOrders::$StartDate, null: ''); ?>" />
8             <label>End Date: <input type="text" class="form-control date-picker" name="end_date" value="<php echo Dates::StandardDate(StateTaxOrders::$EndDate, null: ''); ?>" /></label>
9             <label>State: <?php echo FormClass::SelectItems(StatesClass::GetStates(), StateTaxOrders::$State, id: 'state', class: 'form-control'); ?></label>
10            <input type="submit" class="btn btn-primary" />
11        </form>
12    </div>
13 </div>
14
```

Note that Dates::StandardDate is used to convert the date into a string that the date-picker understands. And we pass an empty string to the null parameter so that it does not default to today's date.

Now that our form is wired up, it is time to start generating the report.

Step 7: Pass the Form Values to the Stored Procedure

Since the stored procedure is in the Ismweb database and named GetStTaxOrders then the name of the class used to execute the stored procedure is ms_sp_IsmwebGetsttaxordersClass.

All stored procedure classes have a GetReport function which takes the parameters that the stored procedure expects. The result is an array of the stored procedure class with the properties fill in based on the columns returned.

```
1 <?php
2
3 /**
4  * Class StateTaxOrders
5  */
6 class StateTaxOrders extends BasePage
7 {
8     public static $StartDate;
9     public static $EndDate;
10    public static $State;
11
12    /* @var $Report ms_sp_IsmwebGetsttaxordersClass[] */
13    public static $Report;
14
15
16    public static function DoInit()
17    {
18        self::$MasterPage = MASTERPAGE_USER_SIDEMENU;
19    }
20
21    public static function DoGet()
22    {
23        self::$State = self::$Request->state;
24        self::$StartDate = Dates::Datestamp(self::$Request->start_date, null: '');
25        self::$EndDate = Dates::Datestamp(self::$Request->end_date, null: '');
26
27        self::$Report = [];
28
29        if(self::$State && self::$StartDate && self::$EndDate) {
30            self::$Report = ms_sp_IsmwebGetsttaxordersClass::GetReport(self::$State, self::$StartDate, self::$EndDate);
31        }
32    }
33 }
34
35 $Web->PageMode = QUICKDRY_MODE_STATIC;
36
37
```

PHPDoc is used to tell the IDE what type a property is. Now when we reference \$Report in the web view file, the IDE will autocomplete the properties for you.

We need to default self::\$Report to an empty array so that we don't get an error later when we try to render the result to the end user.

Step 8: Defining the Properties of the Stored Procedure

There is no way for the framework to know what columns your stored procedure is returning and so it is using a class type which makes it easy to fill in that information later.

Submit the form with valid inputs and you'll get:

```
public $order_no;
public $order_line_no;
public $product_code;
public $Invoice_Date;
public $Taxable_Flag;
public $Actual_Tax_Amount;
public $Base_Billed_Amount;
public $Base_Adjustment_Amount;
public $Base_Writeoff_Amount;
public $Base_Refund_Amount;
public $City;
public $STATE;
public $Postal_Code;
public $County;

#0 Debug::_debug_string_backtrace() called at [C:\web\persephone\htdocs\QuickDRY\utilities\Debug.php:89]
#1 Debug::_Debug(public $order_no;
public $order_line_no;
public $product_code;
public $Invoice_Date;
public $Taxable_Flag;
public $Actual_Tax_Amount;
public $Base_Billed_Amount;
```

That is the PHP code you will paste into the stored procedure class. In the IDE, hold down control and click on `ms_sp_IsmwebGetsttaxordersClass` and you will be taken to the file you need to add this code to.

```

1  <?php
2
3  /**
4   * Class ms_sp_IsmwebGetsttaxordersClass
5   */
6  class ms_sp_IsmwebGetsttaxordersClass extends db_ms_sp_IsmwebGetsttaxordersClass
7  {
8      public $order_no;
9      public $order_line_no;
10     public $product_code;
11     public $Invoice_Date;
12     public $Taxable_Flag;
13     public $Actual_Tax_Amount;
14     public $Base_Billed_Amount;
15     public $Base_Adjustment_Amount;
16     public $Base_Writeoff_Amount;
17     public $Base_Refund_Amount;
18     public $City;
19     public $STATE;
20     public $Postal_Code;
21     public $County;
22
23     /**
24      * ms_sp_IsmwebGetsttaxordersClass constructor.
25      * @param null $row
26      */
27     public function __construct($row = null)
28     {
29         if($row) {
30             $this->HaltOnError( true_or_false: false);
31             $this->FromRow($row);
32             if($this->HasMissingProperties()) {
33                 Halt($this->GetMissingProperties());
34             }
35             $this->HaltOnError( true_or_false: true);
36         }
37     }
38 }

```

Note that any changes made to db_ms_sp_IsmwebGetsttaxordersClass will be overwritten when the code generation tool is run again.

Reload the page and no errors will appear.

Step 9: Render the Results to the User

```
<div class="panel panel-default">
  <div class="panel-heading">
    <div class="panel-title">Report</div>
  </div>
  <div class="panel-body">
    <table class="table table-striped" style="font-size: 0.9em;">
      <thead>
        <tr>
          <th></th>
        </tr>
      </thead>
    </table>
  </div>
</div>
```

Start with the basic outline.

Next, we'll copy and paste the properties into the HTML

```

15 <div class="panel panel-default">
16   <div class="panel-heading">
17     <div class="panel-title">Report</div>
18   </div>
19   <div class="panel-body">
20     <table class="table table-striped" style="font-size: 0.9em;">
21       <thead>
22         <tr>
23           <th></th>
24           public $order_no;
25           public $order_line_no;
26           public $product_code;
27           public $Invoice_Date;
28           public $Taxable_Flag;
29           public $Actual_Tax_Amount;
30           public $Base_Billed_Amount;
31           public $Base_Adjustment_Amount;
32           public $Base_Writeoff_Amount;
33           public $Base_Refund_Amount;
34           public $City;
35           public $STATE;
36           public $Postal_Code;
37           public $County;
38
39         </tr>
40       </thead>
41     </table>
42   </div>
43 </div>
44

```

Now, highlight the properties and replace (ctrl-r) "public \$" with <th> and ; with </th>

Notice the extra th at the top. This is for the counter.


```

<div class="panel panel-default">
  <div class="panel-heading">
    <div class="panel-title">Report</div>
  </div>
  <div class="panel-body">
    <table class="table table-striped" style="font-size: 0.9em;">
      <thead>
        <tr>
          <th></th>
          <th>order_no</th>
          <th>order_line_no</th>
          <th>product_code</th>
          <th>Invoice_Date</th>
          <th>Taxable_Flag</th>
          <th>Actual_Tax_Amount</th>
          <th>Base_Billed_Amount</th>
          <th>Base_Adjustment_Amount</th>
          <th>Base_Writeoff_Amount</th>
          <th>Base_Refund_Amount</th>
          <th>City</th>
          <th>STATE</th>
          <th>Postal_Code</th>
          <th>County</th>
        </tr>
      </thead>
    </table>
  </div>
</div>

```

Now we can fill in the table body with the actual data

```

      <th>County</th>
    </tr>
  </thead>
  <tbody>
    <?php foreach(StateTaxOrders::$Report as $i => $item) { ?>
      <tr>
        <td><?php echo $i + 1; ?></td>
      </tr>
    <?php } ?>
  </tbody>
</table>
</div>

```

Once again, take the properties of the class and paste them in


```
</tr>
</thead>
<tbody>
<?php foreach(StateTaxOrders::$Report as $i => $item) { ?>
    <tr>
        <td><?php echo $i + 1; ?></td>
        public $order_no;
        public $order_line_no;
        public $product_code;
        public $Invoice_Date;
        public $Taxable_Flag;
        public $Actual_Tax_Amount;
        public $Base_Billed_Amount;
        public $Base_Adjustment_Amount;
        public $Base_Writeoff_Amount;
        public $Base_Refund_Amount;
        public $City;
        public $STATE;
        public $Postal_Code;
        public $County;
    </tr>
<?php } ?>
</tbody>
</table>
</div>
</div>
```

Highlight the properties and use ctrl-r to replace "public \$" with "<td><?php echo \$item->"

Replace ";" with "; ?></td>"

```

<tbody>
<?php foreach(StateTaxOrders::$Report as $i => $item) { ?>
    <tr>
        <td><?php echo $i + 1; ?></td>
        <td><?php echo $item->order_no; ?></td>
        <td><?php echo $item->order_line_no; ?></td>
        <td><?php echo $item->product_code; ?></td>
        <td><?php echo $item->Invoice_Date; ?></td>
        <td><?php echo $item->Taxable_Flag; ?></td>
        <td><?php echo $item->Actual_Tax_Amount; ?></td>
        <td><?php echo $item->Base_Billed_Amount; ?></td>
        <td><?php echo $item->Base_Adjustment_Amount; ?></td>
        <td><?php echo $item->Base_Writeoff_Amount; ?></td>
        <td><?php echo $item->Base_Refund_Amount; ?></td>
        <td><?php echo $item->City; ?></td>
        <td><?php echo $item->STATE; ?></td>
        <td><?php echo $item->Postal_Code; ?></td>
        <td><?php echo $item->County; ?></td>
    </tr>
<?php } ?>
</tbody>
</table>
</div>
</div>

```

Notice that we have a date being returned. We need to convert that into a string because it is date object.

```

<td><?php echo $item->order_line_no; ?></td>
<td><?php echo $item->product_code; ?></td>
<td><?php echo Dates::StandardDate($item->Invoice_Date); ?></td>
<td><?php echo $item->Taxable_Flag; ?></td>
<td><?php echo $item->Actual_Tax_Amount; ?></td>

```

Start Date:
End Date:
State:

Report

	order_no	order_line_no	product_code	Invoice_Date	Taxable_Flag	Actual_Tax_Amount	Base_Billed_Amount	Base_Adju
1	1001066991	3	CPSM2_180_EXT	3/8/2018	N	.00	114.50	.00
2	1001066991	4	CPSM3_180_EXT	3/8/2018	N	.00	114.50	.00
3	1001139039	1	340Dues	3/23/2018	N	.00	110.00	.00
4	1001139039	2	DUES	3/23/2018	N	.00	140.00	.00
5	1001139953	1	560Dues	3/27/2018	N	.00	99.00	.00
6	1001139953	2	DUES	3/27/2018	N	.00	140.00	.00
7	1001142516	1	PRORENEW	3/21/2018	N	.00	190.00	.00
8	1001143640	1	PRORENEW	4/17/2018	N	.00	190.00	.00
9	1001166616	1	PRORENEW	4/2/2018	N	.00	190.00	.00
10	1001167126	1	PRORENEW	4/3/2018	N	.00	190.00	.00
11	1001167425	1	PRORENEW	3/19/2018	N	.00	190.00	.00
12	1001167446	1	PRORENEW	3/5/2018	N	.00	190.00	.00
13	1001167451	1	PRORENEW	3/5/2018	N	.00	190.00	.00

Step 10: Allow the User to Export the Report to XLSX

```
<div class="panel panel-default">
  <div class="panel-heading">
    <div class="panel-title">Report <a href="/Accounting/StateTaxOrders?export=xls&start_date=<?php echo StateTaxOrders
  </div>
  <div class="panel-body">
    <table class="table table-striped" style="...">
      <thead>
        <tr>
```

The easiest way to add a download link is to add “export=xls” to the link along with all the Request parameters.

`<i class="fa fa-download"></i>` is used within the A tag which gives you a nice download icon for the user to click.

For pages with more complex reports it may be better to use a form with POST or use the Session object so that the settings do not have to be passed through the URL which has limitations.

Once the link is in place, it is necessary to implement the code that will handle the request for the download.

```
34 public static function DoExportToXLS()
35 {
36     $se = new SimpleExcel();
37     $se->Filename = 'StateTaxOrders.' . self::$StartDate . '.' . self::$EndDate . '.' . self::$State . '.xlsx';
38     $se->Title = 'State Tax Orders';
39     $se->Report = self::$Report;
40     $se->Columns = [
41
42     ];
43     SimpleExcel::ExportSpreadsheet( $se );
44 }
```

The static function DoExportToXLS is processed after the DoGet or DoPost function is executed when the export=xls request parameter is set.

SimpleExcel is used to configure an export. An array of SimpleExport objects can be exported to a single XLSX file but that will be covered elsewhere.

Filename needs to end with .xlsx

The Title must be set and is the name of the worksheet.

Report can be an array of any object type.

Columns defines how to convert the properties of the object into columns in the worksheet.

To then generate the file for the end user, SimpleExcel::ExportSpreadsheet(\$se) is used.

Setting up the columns for the export is very similar to setting up the columns for the table displayed to the user.

```

39     $se->Report = self::$Report;
40     $se->Columns = [
41     public $order_no;
42     public $order_line_no;
43     public $product_code;
44     public $Invoice_Date;
45     public $Taxable_Flag;
46     public $Actual_Tax_Amount;
47     public $Base_Billed_Amount;
48     public $Base_Adjustment_Amount;
49     public $Base_Writeoff_Amount;
50     public $Base_Refund_Amount;
51     public $City;
52     public $STATE;
53     public $Postal_Code;
54     public $County;
55     ];

```

First, copy and past the properties of the stored procedure class to the columns array. Then highlight them.

Replace “public \$” with “new SimpleExcel_Column('aaa',”

Replace ; with '),

```

34 public static function DoExportToXLS()
35 {
36     $se = new SimpleExcel();
37     $se->Filename = 'StateTaxOrders.' . self::$StartDate . '.' . self::$EndDate . '.' . self::$Report;
38     $se->Title = 'State Tax Orders';
39     $se->Report = self::$Report;
40     $se->Columns = [
41         new SimpleExcel_Column( Header: 'aaa', Property: 'order_no'),
42         new SimpleExcel_Column( Header: 'aaa', Property: 'order_line_no'),
43         new SimpleExcel_Column( Header: 'aaa', Property: 'product_code'),
44         new SimpleExcel_Column( Header: 'aaa', Property: 'Invoice_Date'),
45         new SimpleExcel_Column( Header: 'aaa', Property: 'Taxable_Flag'),
46         new SimpleExcel_Column( Header: 'aaa', Property: 'Actual_Tax_Amount'),
47         new SimpleExcel_Column( Header: 'aaa', Property: 'Base_Billed_Amount'),
48         new SimpleExcel_Column( Header: 'aaa', Property: 'Base_Adjustment_Amount'),
49         new SimpleExcel_Column( Header: 'aaa', Property: 'Base_Writeoff_Amount'),
50         new SimpleExcel_Column( Header: 'aaa', Property: 'Base_Refund_Amount'),
51         new SimpleExcel_Column( Header: 'aaa', Property: 'City'),
52         new SimpleExcel_Column( Header: 'aaa', Property: 'STATE'),
53         new SimpleExcel_Column( Header: 'aaa', Property: 'Postal_Code'),
54         new SimpleExcel_Column( Header: 'aaa', Property: 'County'),
55     ];
56     SimpleExcel::ExportSpreadsheet( $se );
57 }

```

You can now go through and copy and paste the property into the header.


```

34 public static function DoExportToXLS()
35 {
36     $se = new SimpleExcel();
37     $se->Filename = 'StateTaxOrders.' . self::$StartDate . '.' . self::$EndDate . '.' . self::$State . '.xlsx';
38     $se->Title = 'State Tax Orders';
39     $se->Report = self::$Report;
40     $se->Columns = [
41         new SimpleExcel_Column( Header: 'order_no', Property: 'order_no'),
42         new SimpleExcel_Column( Header: 'order_line_no', Property: 'order_line_no'),
43         new SimpleExcel_Column( Header: 'product_code', Property: 'product_code'),
44         new SimpleExcel_Column( Header: 'Invoice_Date', Property: 'Invoice_Date'),
45         new SimpleExcel_Column( Header: 'Taxable_Flag', Property: 'Taxable_Flag'),
46         new SimpleExcel_Column( Header: 'Actual_Tax_Amount', Property: 'Actual_Tax_Amount'),
47         new SimpleExcel_Column( Header: 'Base_Billed_Amount', Property: 'Base_Billed_Amount'),
48         new SimpleExcel_Column( Header: 'Base_Adjustment_Amount', Property: 'Base_Adjustment_Amount'),
49         new SimpleExcel_Column( Header: 'Base_Writeoff_Amount', Property: 'Base_Writeoff_Amount'),
50         new SimpleExcel_Column( Header: 'Base_Refund_Amount', Property: 'Base_Refund_Amount'),
51         new SimpleExcel_Column( Header: 'City', Property: 'City'),
52         new SimpleExcel_Column( Header: 'STATE', Property: 'STATE'),
53         new SimpleExcel_Column( Header: 'Postal_Code', Property: 'Postal_Code'),
54         new SimpleExcel_Column( Header: 'County', Property: 'County'),
55     ];
56     SimpleExcel::ExportSpreadsheet( $se );
57 }

```

And finally, you will need to force a couple property types.

```

34 public static function DoExportToXLS()
35 {
36     $se = new SimpleExcel();
37     $se->Filename = 'StateTaxOrders.' . self::$StartDate . '.' . self::$EndDate . '.' . self::$State . '.xlsx';
38     $se->Title = 'State Tax Orders';
39     $se->Report = self::$Report;
40     $se->Columns = [
41         new SimpleExcel_Column( Header: 'order_no', Property: 'order_no', PropertyType: SIMPLE_EXCEL_PROPERTY_TYPE_AS_GIVEN),
42         new SimpleExcel_Column( Header: 'order_line_no', Property: 'order_line_no'),
43         new SimpleExcel_Column( Header: 'product_code', Property: 'product_code'),
44         new SimpleExcel_Column( Header: 'Invoice_Date', Property: 'Invoice_Date', PropertyType: SIMPLE_EXCEL_PROPERTY_TYPE_DATE),
45         new SimpleExcel_Column( Header: 'Taxable_Flag', Property: 'Taxable_Flag'),
46         new SimpleExcel_Column( Header: 'Actual_Tax_Amount', Property: 'Actual_Tax_Amount'),
47         new SimpleExcel_Column( Header: 'Base_Billed_Amount', Property: 'Base_Billed_Amount'),
48         new SimpleExcel_Column( Header: 'Base_Adjustment_Amount', Property: 'Base_Adjustment_Amount'),
49         new SimpleExcel_Column( Header: 'Base_Writeoff_Amount', Property: 'Base_Writeoff_Amount'),
50         new SimpleExcel_Column( Header: 'Base_Refund_Amount', Property: 'Base_Refund_Amount'),
51         new SimpleExcel_Column( Header: 'City', Property: 'City'),
52         new SimpleExcel_Column( Header: 'STATE', Property: 'STATE'),
53         new SimpleExcel_Column( Header: 'Postal_Code', Property: 'Postal_Code'),
54         new SimpleExcel_Column( Header: 'County', Property: 'County'),
55     ];
56     SimpleExcel::ExportSpreadsheet( $se );
57 }

```

AS GIVEN ensures that leading zeros are preserved. DATE formats the date object to just the date. By default, Excel has some weird formatting when you put a date in a column. Those are the two most common types that will be used to help keep exports readable.


Now when you refresh the report you will see

Report 

	order_no	order_line_no	product_code
1	1001143640	1	PRORENEW
2	1001166616	1	PRORENEW
3	1001167126	1	PRORENEW

Clicking the download icon will get you the Excel download

Step 11: Tidy Up

Report 

Order #	Line #	Product Code	Invoice Date	Taxable Flag	Actual Tax Amount	Base Billed Amount	Base Adjustment Amount	Base Writeoff Amount	Base Refund Amount	City	State	Postal Code	County	
1	1001143640	1	PRORENEW	4/17/2018	N	--	\$190.00	--	--	--	Corona	CA	92881	Riverside
2	1001166616	1	PRORENEW	4/2/2018	N	--	\$190.00	--	--	--	Carlsbad	CA	92008	San Diego
3	1001167126	1	PRORENEW	4/3/2018	N	--	\$190.00	--	--	--	Bakersfield	CA	93311	Kern
4	1001170594	2	ISM-18-CHALLENGES	4/25/2018	N	--	--	--	--	--	Napa	CA	94558	Napa
5	1001171425	1	PRORENEW	4/19/2018	N	--	\$190.00	--	--	--	San Dimas	CA	91773	Los Angeles
6	1001172288	21	ISM-18-FULFILL	4/20/2018	N	--	--	--	--	--	Hercules	CA	94547	Contra Costa
7	1001172288	22	ISM-18-PROJECT	4/20/2018	N	--	--	--	--	--	Hercules	CA	94547	Contra Costa

Removing underscores in the header columns and replacing them with spaces allows the page to compress more allowing more columns to fit. The header is often longer than the data. Right aligning numbers and dates helps with readability. And using the `Strings::Currency` function with money formats it in a more readable way. It also converts zero amounts to two dashes which makes them and non-zero values easier to see visually.

```

40
41 <tbody>
42 <?php foreach(StateTaxOrders::$Report as $i => $item) { ?>
43     <tr>
44         <td><?php echo $i + 1; ?></td>
45         <td style="text-align: right"><?php echo $item->order_no; ?></td>
46         <td><?php echo $item->product_code; ?></td>
47         <td style="text-align: right"><?php echo Dates::StandardDate($item->Invoice_Date); ?></td>
48         <td><?php echo $item->Taxable_Flag; ?></td>
49         <td style="text-align: right"><?php echo Strings::Currency($item->Actual_Tax_Amount); ?></td>
50         <td style="text-align: right"><?php echo Strings::Currency($item->Base_Billed_Amount); ?></td>
51         <td style="text-align: right"><?php echo Strings::Currency($item->Base_Adjustment_Amount); ?></td>
52         <td style="text-align: right"><?php echo Strings::Currency($item->Base_Writeoff_Amount); ?></td>
53         <td style="text-align: right"><?php echo Strings::Currency($item->Base_Refund_Amount); ?></td>
54         <td><?php echo $item->City; ?></td>
55         <td><?php echo $item->STATE; ?></td>
56         <td><?php echo $item->Postal_Code; ?></td>
57         <td><?php echo $item->County; ?></td>
58     </tr>
59 <?php } ?>
60 </tbody>

```