How do I create my own resource list?

To create a resource list you need a normal Laravel controller and a class that extends **Aero\Admin\ResourceListsAbstractResourceList**.

```
<?php

namespace Acme\MyModule\ResourceLists;

use Aero\Admin\ResourceLists\AbstractResourceList;

class OrdersResourceList extends AbstractResourceList
{
   //
}
```

Your controller should be like this example but for your new resource list:

```
<?php

namespace Acme\MyModule\Http\Controllers;

use Aero\Admin\Http\Controllers\Controller;
use Aero\Admin\ResourceLists\OrdersResourceList;
use Illuminate\Http\Request;

class OrdersController extends Controller
{
   public function index(OrdersResourceList $list, Request $request)
   {
       return view('admin::resource-lists.index', [
           'list' => $list = $list(),
           'results' => $list->apply($request->all())
```

```
                ->with('status', 'customer', 'items', 'currency')
                ->paginate($request->input('per_page', 24) ?? 24),
        ]);
    }
}
```

Adding Columns to your Resource List

To add columns to your resource list you need to implement the protected **columns()**method and make it return an array of your columns. The entries in the array should use the **Aero\Admin\ResourceLists\ResourceListColumn::create**helper method. This method expects a string parameter for the columns heading, a closure parameter for how to render the columns data (this can return a string), and an optional string parameter for the columns key (by default this is generated from the columns header).

```
<?php

namespace Acme\MyModule\ResourceLists;

use Aero\Admin\ResourceLists\AbstractResourceList;
use Aero\Admin\ResourceLists\ResourceListColumn;

class OrdersResourceList extends AbstractResourceList
{
    protected function columns(): array
    {
        return [
            ResourceListColumn::create('Order', function ($row) {
                return view('admin::resource-lists.link', [
                    'route' => route('admin.orders.view', array_merge(request()->all(),
['order' => $row])),
                    'text' => '#'.$row->reference,
                ])->render();
            }),
        ];
    }
}
```

Making Columns Searchable

You can use the **setSearchAction**method to define how your column is searched. This method accepts a closure (that is passed the query and search term) that defines how the search should happen and an optional string for the text to show in the search field dropdown (this is column header by default).

```
ResourceListColumn::create('Order', function ($row) {
   return view('admin::resource-lists.link', [
       'route' => route('admin.orders.view', array_merge(request()->all(), ['order' =>
$row])),
       'text' => '#'.$row->reference,
   ])->render();
})
->setSearchAction(function ($query, $term) {
   $searchTerm = ltrim($search, '#');

   $query->whereLower('reference', 'like', "%{$searchTerm}%");
}, 'Order Reference')
```

Making Columns Invisible

You can use the **invisible**method to make your column invisible. The main reason for wanting to do this is to add a search for a column that shouldn’t be rendered.

```
ResourceListColumn::create('Order', function ($row) {
   return view('admin::resource-lists.link', [
       'route' => route('admin.orders.view', array_merge(request()->all(), ['order' =>
$row])),
       'text' => '#'.$row->reference,
   ])->render();
})
->invisible()
```

Handling Searching for your Resource List

When your resource list is searched without a specific column being selected a **handleSearch()**function will be called to handle the search. The method will be passed the search term and you are able to use **$this->query**to access the query.

```
<?php

namespace Acme\MyModule\ResourceLists;

use Aero\Admin\ResourceLists\AbstractResourceList;

class OrdersResourceList extends AbstractResourceList
{
   protected function handleSearch($search)
   {
       $searchTerm = ltrim($search, '#');

       $this->query->whereLower('reference', 'like', "%{$searchTerm}%");
   }
}
```

Adding Row Views to your Resource List

Row views are additional rows rendered after each complete row (the best example of this is how the order items are listed after each row of the order on the orders page). To add a row view you simply need to provide your view in the protected **$rowViews**array.

```
<?php

namespace Acme\MyModule\ResourceLists;

use Aero\Admin\ResourceLists\AbstractResourceList;

class OrdersResourceList extends AbstractResourceList
{
   protected $rowViews = [
       'admin::resource-lists.orders.items',
   ];
}
```

Adding Buttons to your Resource List Header

Most resource list pages will require a create button. You can easily add buttons to the top of your resource list page by defining a header slot. Once you have defined a header slot you can inject your view using the admin slots helper.

```
<?php

namespace Acme\MyModule\ResourceLists;

use Aero\Admin\ResourceLists\AbstractResourceList;

class OrdersResourceList extends AbstractResourceList
{
   protected $headerSlot = 'my-list.index.header.buttons';
}
```

Adding Admin Filters to your Resource List

First you will need to create an Admin Filter class as shown in the artile "How do I create a custom admin filter?

To register your newly created admin filter with your resource list you need to ensure that your admin filter class is in the resource lists protected static $filters array.

```
<?php

namespace Acme\MyModule\ResourceLists;

use Aero\Admin\Filters\Order\OrderStatusAdminFilter;
use Aero\Admin\ResourceLists\AbstractResourceList;

class OrdersResourceList extends AbstractResourceList
```

```
{
   protected $filters = [
       OrderStatusAdminFilter::class,
   ];
}
```

Adding Sort Bys to your Resource List

To add sort bys to your resource list you need to implement the protected **sortBys()**method and make it return an array of your sort bys. The entries in the array should use the **Aero\Admin\ResourceLists\ResourceListSortBy::create()**helper method. This method expects an array parameter holding the dropdown options that will be shown in the frontend dropdown, and a closure that will be executed when the sort option is active.

```
<?php

namespace Acme\MyModule\ResourceLists;

use Aero\Admin\ResourceLists\AbstractResourceList;
use Aero\Admin\ResourceLists\ResourceListSortBy;

class OrdersResourceList extends AbstractResourceList
{
   protected function sortBys(): array
   {
       return [
           ResourceListSortBy::create([
               'order-az' => 'Order A to Z',
               'order-za' => 'Order Z to A',
           ], function ($sortBy, $query) {
               return $sortBy === 'order-az' ? $query->orderBy('reference') : $query-
>orderByDesc('reference');
           }),
       ];
   }
}
```

If you would like your sort by to be enabled by default then you call pass null for the first parameter like this:

```
<?php

namespace Acme\MyModule\ResourceLists;

use Aero\Admin\ResourceLists\AbstractResourceList;
use Aero\Admin\ResourceLists\ResourceListSortBy;

class OrdersResourceList extends AbstractResourceList
{
   protected function sortBys(): array
   {
       return [
           ResourceListSortBy::create(null, function ($sortBy, $query) {
               return $query->orderByRaw('coalesce(`orders`.`ordered_at`,
`orders`.`created_at`) desc');
           }),
       ];
   }
}
```

Revision #1
Created 2026-09-02 13:28:56 UTC by Michael Price
Updated 2026-09-02 13:28:56 UTC by Michael Price