

Resource Lists

- [What are the available resource lists?](#)
- [How do I create a custom resource list?](#)
- [How do I add a column to an existing resource list?](#)
- [How do I add a filter to an existing resource list?](#)
- [How do I add a sort by to an existing resource list?](#)
- [How do I create a custom bulk action?](#)
- [How do I add a bulk action to an existing resource list?](#)

What are the available resource lists?

Class	View
<code>\Aero\Admin\ResourceLists\AttributeGroupsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\BlocksResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\CategoriesResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\CollectionsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\CombinationsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\CountriesResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\CountryGroupsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\CurrenciesResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\CustomerGroupsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\CustomersResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\CustomerTaxGroupsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\DeletedProductsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\DiscountsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\ExportLogsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\FlagsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\FulfillmentMethodsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\FulfillmentsResourceList</code>	<code>admin::resource-lists.index</code>

<code>\Aero\Admin\ResourceLists\LocationsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\MailNotificationsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\ManufacturersResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\MissingPagesResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\OrderReturnReasonGroupsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\OrdersResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\OrderStatusesResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\OrderStatusGroupsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\PagesResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\PaymentMethodsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\PriceListsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\ProductsResourceList</code>	<code>admin::catalog.products.index</code>
<code>\Aero\Admin\ResourceLists\ProductTaxGroupsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\ProductTypesResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\RedirectsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\SettingDefinitionsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\ShippingMethodsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\SubscriptionActionsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\SubscriptionPlansResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\SubscriptionsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\TagGroupsResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\TaxRatesResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\TaxRulesResourceList</code>	<code>admin::resource-lists.index</code>

<code>\Aero\Admin\ResourceLists\UsersResourceList</code>	<code>admin::resource-lists.index</code>
<code>\Aero\Admin\ResourceLists\VouchersResourceList</code>	<code>admin::resource-lists.index</code>

How do I create a custom resource list?

To create a resource list you need a normal Laravel controller and a class that extends

`Aero\Admin\ResourceLists\AbstractResourceList`.

```
class OrdersResourceList extends \Aero\Admin\ResourceLists\AbstractResourceList
{
    //
}
```

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

```
class OrdersController extends \Aero\Admin\Http\Controllers\Controller
{
    public function index(\Aero\Admin\ResourceLists\OrdersResourceList $list,
        \Illuminate\Http\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),
        ]);
    }
}
```

Columns

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):

```
class OrdersResourceList extends \Aero\Admin\ResourceLists\AbstractResourceList
{
    protected function columns(): array
    {
        return [
            \Aero\Admin\ResourceLists\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();
            }),
        ];
    }
}
```

Searchable Columns

You can use the `setSearchAction()` method to define how your column is searched:

```
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')
```

Invisible Columns

You can use the `invisible()` method to make a column invisible. This can be useful if wanting to add a search action but not display the relevant data.

Rows

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:

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

Default Search

When your resource list is searched without a specific column being selected a `handleSearch()` method will be called to handle searching.

The method will be passed the search term and you can use `$this->query` to access the query:

```
class OrdersResourceList extends \Aero\Admin\ResourceLists\AbstractResourceList
{
    protected function handleSearch($search)
    {
        $searchTerm = ltrim($search, '#');

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

Buttons

Most resource list pages will require a create button, typically a create resource 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.

```
class OrdersResourceList extends \Aero\Admin\ResourceLists\AbstractResourceList
{
    protected $headerSlot = 'orders.index.header.buttons';
}
```

See [here](#) for how to inject into admin slots.

Filters

To add a filter to your resource list, simply add it to the `$filters` array:

```
class OrdersResourceList extends \Aero\Admin\ResourceLists\AbstractResourceList
{
    protected $filters = [
        \Aero\Admin\Filters\Order\OrderStatusAdminFilter::class,
    ];
}
```

See [here](#) for how to create an existing admin filter, or [here](#) for how to create a custom filter.

Sort Bys

To add sort bys to your resource list you need to implement the protected `sortBy()` 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:

```
class OrdersResourceList extends \Aero\Admin\ResourceLists\AbstractResourceList
{
    protected function sortBy(): array
    {
        return [
            // Default sort by
            \Aero\Admin\ResourceLists\ResourceListSortBy::create(null, function ($sortBy,
                $query) {
```

```

        return $query->orderByRaw('coalesce(`orders`.`ordered_at`,
`orders`.`created_at`) desc');
    }},

    \Aero\Admin\ResourceLists\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');
    }},
    ];
}
}

```

How do I add a column to an existing resource list?

You can use the `addColumn()` method(s) to add a column to an existing resource list:

```
\Aero\Admin\ResourceLists\OrdersResourceList::extend(function
(\Aero\Admin\ResourceLists\OrdersResourceList $list, \Closure $next) {
    // Add it as the first column
    $list->addColumn('Header', function($row) {
        return 'Content';
    });

    // Add it after a specific column
    $list->addColumnAfter('Header', function($row) {
        return $row->reference;
    }, 'customer');

    // Add it before a specific column
    $list->addColumnBefore('Header', function($row) {
        return $row->reference;
    }, 'customer');

    return $next($list);
});
```

If you need to see the keys of the existing columns in order to position your new column appropriately, you can dump them out:

```
\Aero\Admin\ResourceLists\OrdersResourceList::extend(function
(\Aero\Admin\ResourceLists\OrdersResourceList $list, \Closure $next) {
    dd($list->getColumns()->map->key());

    return $next($list);
});
```


How do I add a filter to an existing resource list?

You can use the `addFilter()` method to add a filter to an existing resource list:

```
\Aero\Admin\ResourceLists\OrdersResourceList::extend(function
(\Aero\Admin\ResourceLists\OrdersResourceList $list, \Closure $next) {
    $list->addFilter(\Aero\Admin\Filters\Order\OrderStatusAdminFilter::class);

    return $next($list);
});
```

How do I add a sort by to an existing resource list?

You can use the `addSortBy()` method to add a filter to an existing resource list:

```
\Aero\Admin\ResourceLists\OrdersResourceList::extend(function
(\Aero\Admin\ResourceLists\OrdersResourceList $list, \Closure $next) {
    $list->addSortBy(\Aero\Admin\ResourceLists\ResourceListSortBy::create([
        'latest' => 'Latest',
        'oldest' => 'Oldest',
    ]), function ($sortBy, $query) {
        return $sortBy === 'oldest' ? $query->orderBy('created_at') : $query-
>orderByDesc('created_at');
    }));

    return $next($list);
});
```

How do I create a custom bulk action?

You can create a class that extends `\Aero\Admin\BulkActions\BulkActionJob` and implement a `handle` method:

```
class DeleteProductsBulkAction extends \Aero\Admin\BulkActions\BulkActionJob
    implements \Illuminate\Contracts\Queue\ShouldQueue
{
    protected $list;

    public function __construct(\Aero\Admin\ResourceLists\ProductsResourceList $list)
    {
        $this->list = $list;
    }

    public function handle(): void
    {
        $this->list->items()->each(static function (\Aero\Catalog\Models\Product $product) {
            $product->delete();
            event(new \Aero\Catalog\Events\ProductDeleted($product));
        });
    }
}
```

You should implement `\Illuminate\Contracts\Queue\ShouldQueue` if you want the bulk action to be queued.

How do I add a bulk action to an existing resource list?

```
\Aero\Admin\BulkAction::create(  
    \Aero\Admin\BulkActions\DeleteProductsBulkAction::class,  
    \Aero\Admin\ResourceLists\ProductsResourceList::class  
)  
->title('Delete products')  
->permissions('products.delete')  
->confirm();
```

See [here](#) for how to create a custom bulk action.

Methods

Method	Description
<code>notRunnable()</code>	Removes the <code>run</code> button from the bulk action, intention being you replace this with a custom button.
<code>wideContent()</code>	Displays bulk action view in a wider container.
<code>confirm()</code>	Prompts the user to confirm running the bulk action.
<code>confirmTitle(string \$title)</code>	Sets the title of the confirmation popup. Default is "Are you sure?"
<code>confirmText(string \$text)</code>	Sets text in the confirmation popup. Default is no text.
<code>confirmButtonText(string \$text)</code>	Sets the confirmation popup confirm button text. Default is "Yes".
<code>title(string \$title)</code>	Sets the title of the bulk action.
<code>processingText(string \$text)</code>	Sets the processing text (shown when a bulk action is queued). Default is <code>__('Bulk action is being processed')</code> .
<code>successText(string \$text)</code>	Sets the success text (shown when a bulk action is ran). Default is <code>__('Bulk action has been processed')</code> .

<code>view(string \$view)</code>	Sets a view for the bulk action to redirect to when ran. Allows options to be provided that are required for the bulk action to run.
<code>group(string \$group)</code>	Sets the group of the bulk action, used when they're displayed to order them.