

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');
    }},
    ];
}
}
```

Revision #2

Created 2026-09-23 09:12:23 UTC by Ethan Payne

Updated 2026-10-02 12:45:08 UTC by Ethan Payne