Admin Filters

Admin Filters

- What is an admin filter?
- How do I create a custom admin filter?
- How do I create a checkbox list admin filter?
- How do I create an options date range admin filter?
- How do I create a dropdown admin filter?
- How do I create a date range admin filter?
- How do I create a searchable select admin filter?

What is an admin filter?

Admin filters are the filters that populate the sidebars of reports and resource lists. As the name suggests, they allow users to filter the data they’re seeing through the report or resource list.

All admin filters ultimately extend **Aero\Admin\Filters\AdminFilter**and therefore all have these common methods that you can optionally override:

title Method

This is a public method that returns the title for the filter. This title is displayed on the frontend. By default the title is generated from the class name.

key Method

This is a public method that returns the key for the filter. The key is used for the request parameter for the filter. By default the key is generated from the class name.

You can read about the other methods in the "How do I create a custom Admin Filter?

How do I create a custom admin filter?

To create a custom admin filter you need to create a class that extends **Aero\Admin\Filters\AdminFilter**and implements the handle method.

handle Method

This method is executed on every request made to a report or resource list with your filter and accepts 2 parameters. The first parameter is a **Symfony\Component\HttpFoundation\ParameterBag**, and the second parameter is the query.

render Method

This method defines how the filter is rendered. By default it will return a view (that you can set with the protected **$view**property) and pass in the **options()**and **viewData()**arrays.

stateFields Method

This method returns an array of names for the parameters this filter uses in the request. It’s important that if you use any parameters outside of the default parameter (which uses the **$this->key()**for the name) that you override this method and return them in the array. This method is used to gather all of the fields that should be reset to remove your filter.

options Method

This method returns an array of data that is passed to the view by the render method. If you’re creating a new filter that may be used by many classes it is a good idea to use this method instead of the viewData method so that classes that extend your class can use the viewData method if required.

viewData Method

This method returns an array of data that is passed to the view by the render method.

Here is the code for our date range admin filter implementation:

```
<?php

namespace Aero\Admin\Filters;

use Symfony\Component\HttpFoundation\ParameterBag;

abstract class DropdownAdminFilter extends AdminFilter
{
   protected $view = AdminFilterTypes::DROPDOWN;

   protected function selectedOption()
   {
       return request()->input($this->key());
   }

   protected function options(): array
   {
       return [
           'title' => $this->title(),
           'key' => $this->key(),
           'options' => $this->dropdowns(),
           'selected' => $this->selectedOption(),
       ];
   }

   public function handle(ParameterBag $parameters, $query)
   {
       if (($selected = $parameters->get($this->key())) && $selected != '') {
           $this->handleDropdown($selected, $query);
       }
   }

   abstract protected function handleDropdown($selected, $query);

   abstract protected function dropdowns(): array;
}
```

How do I create a checkbox list admin filter?

To create a checkbox list admin filter you need to create a class that extends **Aero\Admin\Filters\CheckboxListAdminFilter**and implements the **handleCheckboxList**and **checkboxes**methods.

handleCheckboxList Method

This method is executed when any checkboxes are checked and accepts two parameters. The first parameter is an array of the selected items and the second parameter is the query.

checkboxes Method

This method provides the results shown in the rendered checkbox list on the frontend. It needs to return an array that has id, name, and url as keys. The url key should use the **$this->getUrlFor()**helper method to create a url for the id.

```
<?php

namespace Acme\MyModule\Filters;

use Aero\Admin\Filters\CheckboxListAdminFilter;
use Aero\Cart\Models\OrderStatus;

class OrderStatusStateAdminFilter extends CheckboxListAdminFilter
{
   protected function handleCheckboxList(array $selected, $query)
   {
       $query->whereIn('state', $selected);
   }

   protected function checkboxes(): array
   {
       return OrderStatus::query()->distinct('state')->get()->map(function ($status) {
```

```
            return [
                'id' => $status->state,
                'name' => ucwords(implode(' ', explode('_', $status->state))),
                'url' => $this->getUrlFor($status->state),
            ];
        })->toArray();
    }
}
```

How do I create an options date range admin filter?

The options date range admin filter works in the same way as the date range admin filter but displays some dynamic options to the user on the frontend. These dynamic options are especially useful when users save their applied filters.

To create an options date range admin filter you need to create a class that extends **Aero\Admin\Filters\OptionsDateRangeAdminFilter**and implements the **handleDateRange** method.

handleDateRange Method

This method is executed when a date is set and accepts 3 parameters. The first parameter is a Carbon object for the selected start date, the second parameter is a Carbon object for the selected end date, and the third parameter is the query.

You can switch this filter to not use past options (such as yesterday, last 7 days, last 30 days) by setting the **$lastMode**property to false. If this property is false the options will become future options (such as tomorrow, next 7 days, next 30 days).

```
<?php

namespace Aero\Admin\Filters\Order;

use Aero\Admin\Filters\OptionsDateRangeAdminFilter;

class OrderDeliverOnDateAdminFilter extends OptionsDateRangeAdminFilter
{
   protected $lastMode = false;

   protected function handleDateRange($startDate, $endDate, $query)
   {
       $query->whereBetween('deliver_on', [$startDate, $endDate]);
   }
}
```

How do I create a dropdown admin filter?

To create an options date range admin filter you need to create a class that extends Aero\Admin\Filters\DropdownAdminFilter and implements the handleDropdown and dropdowns methods.

handleDropdown Method

This method is executed when a dropdown option is set and accepts 2 parameters. The first parameter is the value selected and the second parameter is the query.

dropdowns Method

This method provides the results shown in the rendered dropdown on the frontend. It needs to return an array that has name and value as keys. If you use an empty string for a value then your handleDropdown method will not be called when that option is selected. This is useful for allowing users to effectively turn your filter off.

```
<?php

namespace Aero\Admin\Filters\MailNotification;

use Aero\Admin\Filters\DropdownAdminFilter;

class MailNotificationLayoutAdminFilter extends DropdownAdminFilter
{
   protected function handleDropdown($selected, $query)
   {
       switch ($selected) {
           case 'system':
               $query->where('layout', 'system');
               break;
           case 'customer':
               $query->where('layout', 'customer');
```

```
                break;
        }
    }

    protected function dropdowns(): array
    {
        return [
            [
                'value' => '',
                'name' => 'View All',
            ],
            [
                'value' => 'customer',
                'name' => 'Customer',
            ],
            [
                'value' => 'system',
                'name' => 'System',
            ],
        ];
    }
}
```

How do I create a date range admin filter?

To create a date range admin filter you need to create a class that extends **Aero\Admin\Filters\DateRangeAdminFilter**and implements the **handleDateRange**method.

handleDateRange Method

This method is executed when a date is set and accepts 3 parameters. The first parameter is a Carbon object for the selected start date, the second parameter is a Carbon object for the selected end date, and the third parameter is the query.

```
<?php

namespace Acme\MyModule\Filters;

use Aero\Admin\Filters\DateRangeAdminFilter;

class CreatedAtAdminFilter extends DateRangeAdminFilter
{
   protected function handleDateRange($startDate, $endDate, $query)
   {
       $query->whereBetween('created_at', [$startDate, $endDate]);
   }
}
```

How do I create a searchable select admin filter?

To create an options date range admin filter you need to create a class that extends Aero\Admin\Filters\SearchableSelectAdminFilter and implements the handleSearchableSelect, model, modelName, and searchRoute methods.

handleSearchableSelect Method

This method is executed when an option has been selected and accepts 2 parameters. The first parameter is an array of selected values and the second parameter is the query.

model Method

This method needs to return an instance of the model that will be used for searching. This will be used to automatically fetch the selected data.

modelName Method

This method defines the name that should be used in the searchable select for the selected options.

searchRoute Method

This method defines the route that will be used by the searchable select component to fetch data.

```
<?php

namespace Aero\Admin\Filters\Order;

use Aero\Admin\Filters\SearchableSelectAdminFilter;
use Aero\Cart\Models\Discount;
use Illuminate\Database\Eloquent\Model;
```

```
class OrderDiscountAdminFilter extends SearchableSelectAdminFilter
{
   protected $multiple = true;

   protected function handleSearchableSelect(array $selected, $query)
   {
       $query->where(function ($query) use ($selected) {
           $query->whereHas('discounts', static function ($query) use ($selected) {
               $query->whereIn('id', $selected);
           });
       });
   }

   protected function model(): Model
   {
       return new Discount();
   }

   protected function modelName(Model $model)
   {
       return $model->code ?? ($model->name ?? 'Discount #'.$model->id);
   }

   protected function searchRoute(): string
   {
       return route('admin.discounts.search');
   }
}
```