

How do I create a new event?

To create a new event, we first have to scaffold the class using Artisan in the project root directory:

```
php artisan make:event Registered
```

The above command scaffolds a class into the **app/Events** and we can modify it accordingly to our needs.

After applying all the necessary functionality to the event, we have to also create a listener to listen to an event happening. In order for Aero to recognize the event, we have to specify it in a **Service Provider** of choice, for example, a module **Service Provider**:

```
protected $listen = [  
    Registered::class => [  
        // insert your listener here  
    ]  
];
```

Extending ManagedEvent

Aero's **ManagedEvent** is an **optional class** made specifically for Aero events and is used mainly for mailing events. These are a special type of event as all the events that extend **ManagedEvent** are actions.

It's important to note that the listener that listens to an event extending **ManagedEvent**, must also extend the **ManagedListener** class.

Variables

With **ManagedEvent**, we can create variables that can then be accessible in a mailing template. In essence, variables create helper text and allow the user to use data from the event itself. To define some variables, we have to create a property 'variables' in our event:

Adding variables

```
public static $variables = [  
    'product.slug',  
    'product.name',  
    'product.model',  
    'product.summary',  
    'product.description',  
    'product.thumbnail',  
    'product.heading',  
    'product.url',  
    'product.categories.*.name',  
    'product.categories.*.has_featured_image',  
    'product.categories.*.featured_image_file',  
    'product.manufacturer.name',  
    'product.manufacturer.has_logo',  
    'product.manufacturer.logo_file',  
    'product.attributes.*.name',  
    'product.attributes.*.image_file',  
    'product.images.*.image_file',  
    'product.all_images.*.image_file',  
    'product.variants.*.sku',  
    'product.variants.*.has_stock',  
    'product.highest_price',  
    'product.lowest_price',  
    'product.has_reductions',  
];
```

The above variables will then be accessible in our mail template.

An alternative method of adding variables:

```
EventClass::addVariable('product.lowest_price')
```

Or multiple, as an array:

```
EventClass::addVariables(['product.manufacturer.name', 'product.lowest_price'])
```

Revision #1

Created 2026-09-02 13:27:52 UTC by Michael Price

Updated 2026-09-02 13:27:52 UTC by Michael Price