Hooks & Events

Hooks & Events

Jonathan Daggerhart

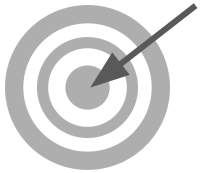
- **Developer at Hook 42**
- **Organizer for Drupal Camp Asheville**

Drupal.org: daggerhart
Twitter: @daggerhart
Blog: https://www.daggerhart.com

Drupal Camp Asheville
Site: https://drupalasheville.com
Twitter: @drupalasheville

Hooks & Events

What we will cover



1. Event Systems - General Overview
2. Hooks in Drupal 7 & 8
3. YAML & Annotations
4. Events in Drupal 8

Hooks & Events

Event Systems - Overview

An **Event System** is a programming pattern that allows a complex system to be easy to extend.
An Event System is made up of the following concepts:

Event Subscribers	Sometimes called “Listeners”, are callable methods or functions that react to an event being propagated throughout the Event Registry.
Event Dispatcher	The mechanism in which and event is “dispatched” throughout the system.
Subscriber Registry	Where event subscribers are collected and sorted. Commonly within a dispatcher.
Event Context	Many events require specific set of data that is important to the subscribers to an event. This can be as simple as a value passed to the Event Subscriber, or as complex as a specially created class that contains the relevant data.

Hooks & Events

Event Systems - Overview cont...

Many complex systems you may already be familiar with have event systems, though they may call their events something else.

- **Drupal** - Hooks & Alters
  - `hook_menu()`
  - `hook_form_alter(&$form, &$form_state, $form_id)`
- **WordPress** - Actions & Filters
  - `add_action('init', function(){});`
  - `add_filter('the_content', function(){});`
- **JavaScript** - Events
  - `element.addEventListener('click', function(){});`

Hooks & Events

Drupal Hooks = Event System

Drupal hooks are registered in the system by defining a function with a specific name. For example, if you want to subscribe to the made up "***`hook_my_event_name`***" event, you must define a new function named **`myprefix_my_event_name()`**, where "*myprefix*" is the name of your module or theme.

Event Subscriber

Event Dispatcher

```php
/**
 * Implements hook_preprocess_page().
 *
 * @param $variables
 */
function example_preprocess_page(&$variables){

}

// Dispatch the hook.
module_invoke_all('preprocess_page', $context);
```

Event Context is provided by dispatcher.

Hooks & Events

WordPress Hooks = Event System

WordPress hooks are registered with the system by using the `add_action()` or `add_filter()` functions to assign another function to an event name.

Event Subscriber

Register Event

Event Dispatcher

```php
/**
 * Function that is registered with the event system.
 *
 * @param $content
 */
function my_event_subscriber($content) {

}
// Register the hook/event.
add_action('the_content', 'my_event_subscriber');


// Dispatch the hook/event.
do_action('the_content', $context);
```

Event Context is provided by dispatcher.

Hooks & Events

Drupal 8 (Symfony) Event Subscribers

In Drupal 8 event s subscribers are registered in a module's **`services.yml`** file as a service with a specific tag.

`my_events/my_events.services.yml`

```yaml
services:
  # My service name. Doesn't need to match anything.
  my_event_subscriber:
    # Fully namespaced class name for the subscriber.
    class: \Drupal\my_events\MyEventSubscriber
    # Tagged as an event_subscriber to register this
    # subscriber with the event_dispatch service.
    tags:
      - { name: 'event_subscriber' }
```

Subscribers are classes which implement the **`EventSubscriberInterface`**

`my_events/src/MyEventSubscriber.php`

```php
class MyEventSubscriber implements EventSubscriberInterface {

  public function getSubscribedEvents() {
    return [
      // Constant => method on this class.
      SOME_EVENT_NAME => 'someMethodOnThisClass',
    ];
  }

  public function someMethodOnThisClass() {}
}
```

Hooks & Events

Drupal 8 Event Context & Dispatcher

In Drupal 8 an event's context is an instance of a class that extends Symfony's **`Event`** class.

```php
class MyEvent extends Event {

  /**
   * Some event classes store their event name
   * strings as constants within the class.
   */
  const MY_EVENT_NAME = 'some_unique_string';

  /**
   * The context for the event is stored on
   * the instance of the event class.
   */
  public $someValue;

  /**
   * Context values are often passed into the
   * constructor.
   *
   * @param $some_value
   */
  function __construct($some_value) {
    $this->someValue = $some_value;
  }
}
```

And the event dispatcher is an instance of Symfony's **`EventDispatcher`** class, made available as a service.

```php
$event_context = new MyEvent('Some arbitrary value.');

$event_dispatcher = \Drupal::service('event_dispatcher');
$event_dispatcher->dispatch(MyEvent::MY_EVENT_NAME, $event_context);
```

Expect To Use Both Hooks & Events

There are relatively very few Events currently in Drupal 8 core, but plenty of contributed modules make use of them. Events are (hopefully) the future in Drupal, and while many Hooks have been replaced by YAML & Annotations, Hooks in the current sense may never be gone.

Notes:

1. When work on a Drupal 8 module, expect to use both Hooks & Events.
2. If you have to decide between dispatching Hooks and Events within your modules, lean towards dispatching an Event instead of a Hook.