DUBLIN
DRUPALCON

DUBLIN
DRUPALCON

The State of Hooking into Drupal

Track: Symfony

The State of Hooking into Drupal

who am I?

Nida Ismail Shah

- Developer at Acquia
- drupal.org: nidaismailshah
- twitter: @nidaismailshah
- nidashah.com

Acquia

Overview

- Hooks - What, Why, When and How.
- State - Where we are, What about hooks? What about events?
- Events/Symfony Event Dispatcher - What, Why, When, and How
- Hooks in D8.

Transition

- Drupal 7 to Drupal 8
  - Procedural to Object Oriented
  - Symfony components.
  - Modularity
  - Clean code
  - Developer experience

what has changed

- Dependency Injection
- Event Subscribers for Hooks
- write portable code
- use code from somewhere else
- Unit testing. objects can be mocked, but how do u mock a global function?

Hooks

“The idea is to be able to run random code at given places in the engine. This random code should then be able to do whatever needed to enhance the functionality. The places where code can be executed are called “*hooks*” and are defined by a fixed interface.” - ***Dries Buytaert.***

… hooks

- “To extend Drupal, a module need simply implement a hook. When Drupal wishes to allow intervention from modules, it determines which modules implement a hook and calls that hook in all enabled modules that implement it.”
- Allow interaction with your module (or even core).
- triggered by function naming convention matching on the hook_name event

… hooks - types

- In common practice, there are a lot of types of hooks that you want to create:
  - **Info hooks:** declare metadata.
  - **Alter hooks**: a common way to edit the contents of a particular object or variable by getting variables in the hook by reference, typically by using **drupal_alter()**

Creating hooks

- // Calling all modules implementing ‘hook_name':
  module_invoke_all(‘name');

- // Calling a particular module's 'hook_name' implementation:
  module_invoke('module_name', ‘name’);

Creating hooks

-
  ```
  // Calling all modules implementing hook_hook_name and
  // Returning results than pushing them into the $result array:
  foreach (module_implements('hook_name') as $module) {
    $result[] = module_invoke($module, 'hook_name');
  }
  ```

-
  ```
  // Calling all modules implementing hook_my_data_alter():
  drupal_alter('my_data', $data);
  ```

module_invoke() D7

```php
/**
 * Invokes a hook in a particular module.
 *
 * All arguments are passed by value. Use drupal_alter() if you need to pass
 * arguments by reference.
 *
 *
 * @return
 *   The return value of the hook implementation.
 */
function module_invoke($module, $hook) {
  $args = func_get_args();
  // Remove $module and $hook from the arguments.
  unset($args[0], $args[1]);
  if (module_hook($module, $hook)) {
    return call_user_func_array($module . '_' . $hook, $args);
  }
}
```

drupal_alter() D7

```php
function drupal_alter($type, &$data, &$context1 = NULL, &$context2 = NULL, &$context3 =
NULL) {
 . . . . .
  // Some alter hooks are invoked many times per page request, so statically
  // cache the list of functions to call, and on subsequent calls, iterate
  // through them quickly.
  if (!isset($functions[$cid])) {
    $functions[$cid] = array();
    $hook = $type . '_alter';
    $modules = module_implements($hook);
    if (!isset($extra_types)) {
      // For the more common case of a single hook, we do not need to call
      // function_exists(), since module_implements() returns only modules with
      // implementations.
      foreach ($modules as $module) {
        $functions[$cid][] = $module . '_' . $hook;
      }
    }
 . . . . .
}
```

order of execution D7

```
function module_implements($hook, $sort = FALSE, $reset = FALSE) {
  . . . . .
  if ($hook != 'module_implements_alter') {
    // Remember the implementations before hook_module_implements_alter().
    $implementations_before = $implementations[$hook];
    drupal_alter('module_implements', $implementations[$hook],
  $hook);
    // Verify implementations that were added or modified.
    foreach (array_diff_assoc($implementations[$hook], $implementations_before) as $module =>
  $group) {
      // If drupal_alter('module_implements') changed or added a $group, the
      // respective file needs to be included.
      if ($group) {
        module_load_include('inc', $module, "$module.$group");
      }
      // If a new implementation was added, verify that the function exists.
      if (!function_exists($module . '_' . $hook)) {
        unset($implementations[$hook][$module]);
      }
    . . . .
  }
```

State

- All the info hooks are gone
  - Replaced with annotations, yaml files etc eg hook_block_info
- hook_init, hook_exit, hook_boot are gone.
- Alter hooks are still there
  - Will likely be replaced with Event Listeners
- [drupal.org]() issue.

Whats wrong? / Why change?

- D7 - Block hooks
  - _info()
  - _view()
  - _configure()
  - _save()
- D8 - Block Plugin
  - build()
  - blockForm()
  - blockSubmit()

… whats wrong? / why change?

- Object oriented.
- Highly modular code.
- Unit tests.
- Decoupling
- Code reuse

Events vs Hooks

- Object oriented.
- Easy to write extensible code.
- Stopping propagation.
- Fire twice on the one event.
- Loosely coupled
- Reusability
- Services

- Procedural
- Ambiguity
- No proper control over hook invoking.
- Tight coupling
- Poor reuse/ duplication

Events

"… an event is an action or an occurrence recognised by software that may be handled by software. "

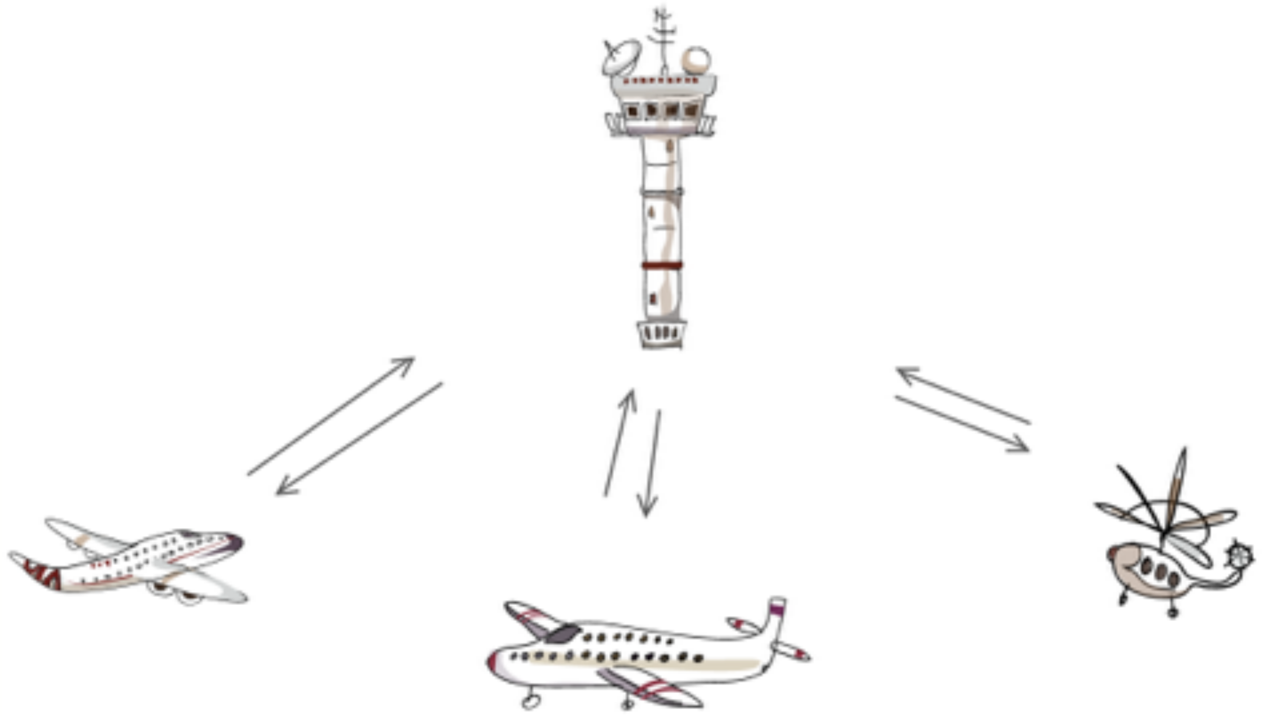
Events

- Events are part of the Symfony framework: they allow for different components of the system to interact and communicate with each other.
- Object oriented way of interaction with core and other modules.
- Mediator Pattern
- Container Aware dispatcher

...

- Code extensibility
- Services can be used within events.
- Inject services into Event Subscribers.
- Services can trigger events too.

Mediator Pattern



Mediator Pattern

- Intent
  - Define an object that encapsulates how a set of objects interact. Mediator promotes loose coupling by keeping objects from referring to each other explicitly, and it lets you vary their interaction independently.
  - Design an intermediary to decouple many peers.

Creating and subscribing to events

What do we need

- Logic
- Dispatch the Event
- Listen to the Event
- Do something.

How do we listen/subscribe to an event

- Implement EventSubscriberInterface in your subscriber class
- Implement the getSubscribedEvents() method
- Write the methods that respond to events to extend functionality
- Declare the Event subscriber as a service. Add to the services.yml

Event subscriber interface

```php
interface EventSubscriberInterface
{
    /**
     * Returns an array of event names this subscriber wants to listen to.
     *
     * The array keys are event names and the value can be:
     *
     *  * The method name to call (priority defaults to 0)
     *  * An array composed of the method name to call and the priority
     *  * An array of arrays composed of the method names to call and respective
     *    priorities, or 0 if unset
     *
     * For instance:
     *
     *  * array('eventName' => 'methodName')
     *  * array('eventName' => array('methodName', $priority))
     *  * array('eventName' => array(array('methodName1', $priority), array('methodName2')))
     *
     * @return array The event names to listen to
     */
    public static function getSubscribedEvents();
}
```

event subscriber class

```
class ConfigFactory implements ConfigFactoryInterface, EventSubscriberInterface {
. . . .

  /**
   * {@inheritdoc}
   */
  static function getSubscribedEvents() {
    $events[ConfigEvents::SAVE][] = array('onConfigSave', 255);
    $events[ConfigEvents::DELETE][] = array('onConfigDelete', 255);
    return $events;
  }
. . . .

}
```

services.yml

```yaml
config.factory:
  class: Drupal\Core\Config\ConfigFactory
  tags:
    - { name: event_subscriber }
    - { name: service_collector, tag: 'config.factory.override', call: addOverride }
  arguments: ['@config.storage', '@event_dispatcher', ‘@config.typed']


services:
  event_demo.alter_response:
    class: Drupal\event_demo\EventSubscriber\AlterResponse
    arguments: [ '@logger.factory' ]
    tags:
      - { name: event_subscriber }
```

callable to perform action

```
class AlterResponse implements EventSubscriberInterface {
  public function loggingDemo(GetResponseEvent $event) {
      // do something.
    }

    public static function getSubscribedEvents() {
      return [
        KernelEvents::REQUEST => 'loggingDemo',
      ];
    }
}
```

order of execution - set priority

```
class AlterResponse implements EventSubscriberInterface {
  public function loggingDemo(GetResponseEvent $event) {
      // do something.
    }

    public static function getSubscribedEvents() {
      return [
        KernelEvents::REQUEST => [‘loggingDemo’, 10],
        KernelEvents::RESPONSE => [
            [‘loggingDemoResp’, 10],
            [‘somethingElse’, 20],
      ];
    }
}
```

Listen

- Define a service in your module, tagged with **‘event_subscriber'**
- Define a class for your subscriber service that implements **\Symfony\Component\EventDispatcher\EventSubscriberInterface**
- Implement the **getSubscribedEvents()** method and return a list of the events this class is subscribed to, and which methods on the class should be called for each one.
- Write the methods that respond to the events; each one receives the event object provided in the dispatch as its one argument.

How do we Dispatch

- Static Event class
- Extend the Event class
- Dispatch the Event

Static Event Class

```php
namespace Symfony\Component\HttpKernel;

/**
 * Contains all events thrown in the HttpKernel component.
 *
 * @author Bernhard Schussek <bschussek@gmail.com>
 */
final class KernelEvents
{
    const REQUEST = 'kernel.request';

    const EXCEPTION = 'kernel.exception';

    const VIEW = 'kernel.view';

    const CONTROLLER = 'kernel.controller';

    const RESPONSE = 'kernel.response';

    const TERMINATE = 'kernel.terminate';

    const FINISH_REQUEST = 'kernel.finish_request';
}
```

Event class

```php
class Event
{
  private $propagationStopped = false;
  private $dispatcher;
  private $name;
  public function isPropagationStopped() {
    return $this->propagationStopped;
  }
  public function stopPropagation() {
    $this->propagationStopped = true;
  }
  public function getName() {
    return $this->name;
  }
  public function setName($name) {
    $this->name = $name;
  }
}
```

“The base Event class provided by the EventDispatcher component is deliberately sparse to allow the creation of API specific event objects by inheritance using OOP. This allows for elegant and readable code in complex applications.”

Extend the generic Event Class

```
class EventDemo extends Event {

  protected $config;


  public function __construct(Config $config) {
    $this->config = $config;
  }

   public function getConfig() {
    return $this->config;
  }

   public function setConfig($config) {
    $this->config = $config;
  }

}
```

Dispatch your event - with logic

```
$dispatcher = \Drupal::service('event_dispatcher');

$event = new EventDemo($config);

$event = $dispatcher->dispatch(EVENT_NAME, $event);
// Alter the data in the Event class.

// Fetch the data from the event.
```

Dispatch

- To dispatch an event, call the **\Symfony\Component\EventDispatcher\EventDispatchInterface::dispatch()** method on the 'event_dispatcher' service.

- The first argument is the unique event name, which you should normally define as a constant in a separate static class.

- The second argument is a **\Symfony\Component\EventDispatcher\Event** object; normally you will need to extend this class, so that your event class can provide data to the event subscribers.

other event dispatchers and listeners

- Event Object
- Generic Event Object
- Container Aware Event Dispatcher
  - Use services within your events
- EventDispatcher Aware Events and Listeners
  - dispatch other events from within the event listeners

What’s happening in core

- ConfigEvents::DELETE, IMPORT, SAVE, RENAME etc
- EntityTypeEvents::CREATE, UPDATE, DELETE
- FieldStorageDefinitionEvents::CREATE, UPDATE, DELETE
- ConsoleEvents::COMMAND, EXCEPTION, TERMINATE

What’s happening in core

- KernelEvents::CONTROLLER, EXCEPTION, REQUEST, RESPONSE, TERMINATE, VIEW

- MigrateEvents:: MAP_DELETE, MAP_SAVE, POST_IMPORT, POST_ROLLBACK, POST_ROW_DELETE, POST_ROW_SAVE,

- RoutingEvents::ALTER, DYNAMIC, FINISHED

Hooks in D8

Hooks in D8

- Some are gone
  - No hook_init
  - No hook_boot
  - No hook_exit, block hooks
- Some are still there
  - form_alter
  - *_alter

Hooks in D8

```
function user_login_finalize(UserInterface $account) {
  \Drupal::currentUser()->setAccount($account);
  \Drupal::logger('user')->notice('Session opened for %name.', array('%name' =>
$account->getUsername()));
  // Update the user table timestamp noting user has logged in.
  // This is also used to invalidate one-time login links.
  $account->setLastLoginTime(REQUEST_TIME);
  \Drupal::entityManager()
    ->getStorage('user')
    ->updateLastLoginTimestamp($account);

  // Regenerate the session ID to prevent against session fixation attacks.
  // This is called before hook_user_login() in case one of those functions
  // fails or incorrectly does a redirect which would leave the old session
  // in place.
  \Drupal::service('session')->migrate();
  \Drupal::service('session')->set('uid', $account->id());
  \Drupal::moduleHandler()->invokeAll('user_login', array($account));
}
```

Hooks in D8

```
/**
 * {@inheritdoc}
 */
public function invoke($module, $hook, array $args = array()) {
  if (!$this->implementsHook($module, $hook)) {
    return;
  }
  $function = $module . '_' . $hook;
   return call_user_func_array($function, $args);
}
```

Hooks in D8

```
/**
 * {@inheritdoc}
 */
public function invokeAll($hook, array $args = array()) {
  $return = array();
  $implementations = $this->getImplementations($hook);
  foreach ($implementations as $module) {
    $function = $module . '_' . $hook;
     $result = call_user_func_array($function, $args);
    if (isset($result) && is_array($result)) {
      $return = NestedArray::mergeDeep($return, $result);
    }
    elseif (isset($result)) {
      $return[] = $result;
    }
  }

  return $return;
}
```

Hooks in D8 - drupal_alter()

```
/**
 * Passes alterable variables to specific hook_TYPE_alter() implementations.
 *
 * This dispatch function hands off the passed-in variables to type-specific
 * hook_TYPE_alter() implementations in modules. It ensures a consistent
 * interface for all altering operations.
 *
 * A maximum of 2 alterable arguments is supported. In case more arguments need
 * to be passed and alterable, modules provide additional variables assigned by
 * reference in the last $context argument
 */
public function alter($type, &$data, &$context1 = NULL, &$context2 = NULL);
```

Hooks in D8 - drupal_alter()

```
protected function buildImplementationInfo($hook) {

   // Allow modules to change the weight of specific implementations, but avoid
  // an infinite loop.
  if ($hook != 'module_implements_alter') {
    // Remember the original implementations, before they are modified with
    // hook_module_implements_alter().
    $implementations_before = $implementations;
    // Verify implementations that were added or modified.
    $this->alter('module_implements', $implementations, $hook);


  }
  return $implementations;
}
```

Hooks in D8 - drupal_alter()

```
\Drupal::moduleHandler->alter($type, $data);

     $this->moduleHandler->alter('views_data', $data);
```

Summary

Summary

- Writing your own module?
  - Log everything and trigger an Event for everything.
- Interact with or alter core?
  - subscribe to an event (if one is fired).
  - Hooks … you don't have too many options.
- Configuration, Admin forms?
  - Plugins
- Simple Extensions
  - Tagged services

simple demo

DUBLIN
DRUPALCON

JOIN US FOR CONTRIBUTION SPRINTS

First Time Sprinter Workshop - 9:00-12:00 - Room Wicklow2A

Mentored Core Sprint - 9:00-18:00 - Wicklow Hall 2B

General Sprints - 9:00 - 18:00 - Wicklow Hall 2A

DUBLIN
DRUPALCON

WHAT DID YOU THINK?

Evaluate This Session

events.drupal.org/dublin2016/sessions/state-hooking-drupal

THANK YOU!

DUBLIN
DRUPALCON

Questions?

Thank You

Acquia