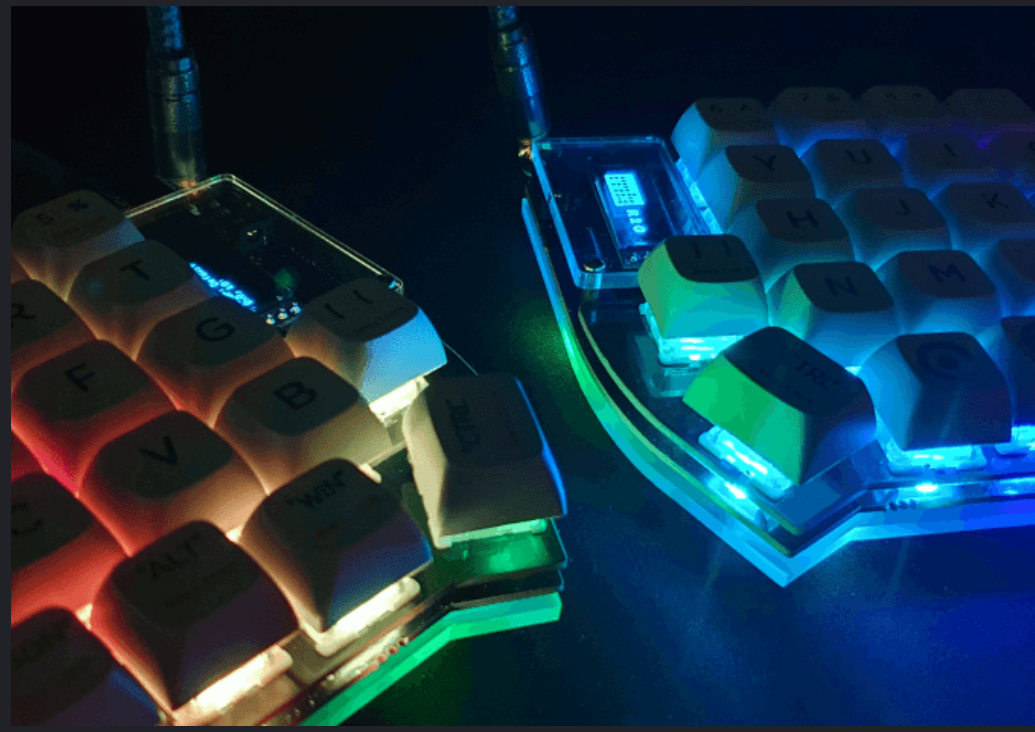
The Drupal Queue API

Philip Norton

Philip Norton

- Developer at Code Enigma
- Writer at `#! code` (www.hashbangcode.com)
- NWDUG co-organiser



Source Code

- Talk is available at: https://github.com/hashbangcode/drupal-queue-api-talk
- All code seen can be found at: https://bit.ly/4gTYaLP
- I have also written about the Queue API on www.hashbangcode.com

The Drupal Queue API

What Is A Queue?

A queue is a "first in, first out" (or FIFO) data model where items are placed into a list and processed in order.

The first item added is processed first, just like a normal queue.

This is a common structure in programming when processing data.

What Is The Queue API?

The Queue API creates a queue data structure and allows tasks to be processed asynchronously. This is normally in order to prevent timeout errors or memory problems.

The Queue API

- The API is highly extensible. By default we have a database queue and a memory queue interface.
- When creating a queue it is common for the database queue to be used.
- The cron system automatically picks up database queue items and processes them.

The Queue API

- The Queue API in Drupal has no user interface as everything happens behind the scenes.
- The Batch API is built upon the Queue API and uses both the database and memory queues. The Batch API has an interface that is shown to users when processing tasks.

The Queue API

The API consists of three parts:

- A storage interface for the queue.
- The queue worker that will process a single item in the queue.
- A processing step that picks items out the queue and processes it using the worker.

When To Use A Queue

- Use a queue when you need to process something that can be done asynchronously. For example:
  - Perform an action on multiple items of content.
  - Synchonise data to (or from) an API.

When To Use A Queue

For example, let's say you wanted to delete all taxonomy items on a site.

Instead of doing this in one go you would add all of the taxonomy IDs to a queue and process them one by one.

Create A Queue

Create A Queue

- To create a queue we use the `queue` service, which is a factory that produces queue objects.

```
/** @var \Drupal\Core\Queue\QueueFactoryInterface $queueFactory */
$queueFactory = \Drupal::service('queue');
```

Create A Queue

- We can use the factory to generate a queue with an arbritary name.

```php
/** @var \Drupal\Core\Queue\QueueInterface $queue */
$queue = $this->queueFactory->get('queue_simple_example');
```

- By default we use the database queue, which stores queue items in the `queue` table.

Create A Queue

- This can be simplified in a single step.

```
/** @var \Drupal\Core\Queue\QueueInterface $queue */
$queue = \Drupal::queue('queue_simple_example');
```

Create A Queue

- Using the queue interface we can then add items to the queue for later processing.

```
$item = new \stdClass();
$item->id = 123;
$queue->createItem($item);
```

- After running this code we will now have the data "123" stored in the queue table in a PHP serialised format.

Create A Queue

- Here is the data in the queue table.

```
> select * from queue;
+---------+--------------------------------+-------------------------------------+--------+------------+
| item_id | name                           | data                                | expire | created    |
+---------+--------------------------------+-------------------------------------+--------+------------+
|    5156 | queue_simple_example           | O:8:"stdClass":1:{s:2:"id";i:123;} |      0 | 1735664606 |
+---------+--------------------------------+-------------------------------------+--------+------------+
```

Create A Queue

- You can see the number of items in your queue quite easily.

```
$queue->numberOfItems()
```

Create A Queue Worker

A queue worker is required to process the tasks in the queue.

Create A Queue Worker

- A queue worker is a plugin that accepts single items from the queue.
- When Drupal is processing the queue it will pass each item to the processor.
- If everything is processed correctly then the item will be deleted from the queue.

Create A Queue Worker

- To define a queue worker we create a class in the directory `src/Plugin/QueueWorker` in a custom module. This contains the following annotation.

```
/**
 * Queue worker for the queue_simple_example.
 *
 * @QueueWorker(
 *   id = "queue_simple_example",
 *   title = @Translation("Queue worker for the simple queue example."),
 *   cron = {"time" = 60}
 * )
 */
class QueueExampleWorker extends QueueWorkerBase
}
```

Create A Queue Worker

- The class must contain a `process()` method, that accepts the data we want to process.

```
public function processItem($data) {
  // $data->id contains the id we added to the queue.
}
```

- If we return nothing from the process method then the queue item is deleted from the queue.

Throwing Exceptions During Processing

We can effect the queue in different ways by throwing different exceptions. Drupal accepts 4.

- `\Drupal\Core\Queue\DelayedRequeueException` - For database queues, the item is added back to the queue and held back for an hour.
- `\Drupal\Core\Queue\RequeueException` - The item is added back into the queue and will be picked up later in the processing.

Throwing Exceptions During Processing

- `\Drupal\Core\Queue\SuspendQueueException` - The item is added back into the queue and the queue execution is stopped.
- `\Exception` - The item is added back into the queue for later processing and an error is logged.

Processing A Queue

Processing A Queue

The easiest way to process a queue is to run cron.

- Via the cron processor form at "/admin/config/system/cron".
- Via the "Run cron" link on the status page.
- Via Drush using the command `drush cron`.

Processing A Queue

- It is possible to process queues outside of cron.
- Use the `queue` factory to load the queue and the queue worker plugin to process the item.

```
$queue = \Drupal::service('queue')->get('queue_simple_example');
$worker = \Drupal::service('plugin.manager.queue_worker')->createInstance('queue_simple_example');

$item = $queue->claimItem();
$worker->processItem($item->data);
$queue->deleteItem($item);
```

Examples Of Queue API In Action

Some live demos!

Run A Queue

- A look at the Queue API.

Custom Queue Processing

- Custom queue processing.

Queue Storage Customisation

- Change the database handler for queues.

Some Tips On Queue API Usage

Top Tips

- Keep an eye on the numbers in your queue.
- Think about the amount of time that it would take to process your queue. Is the cron handler running often enough to process the data?
- For large queues, consider storing a (small) array of items in the data. This helps with the speed of processing.
- Don't store too much data in your queue items, just enough to provide context.

Top Tips

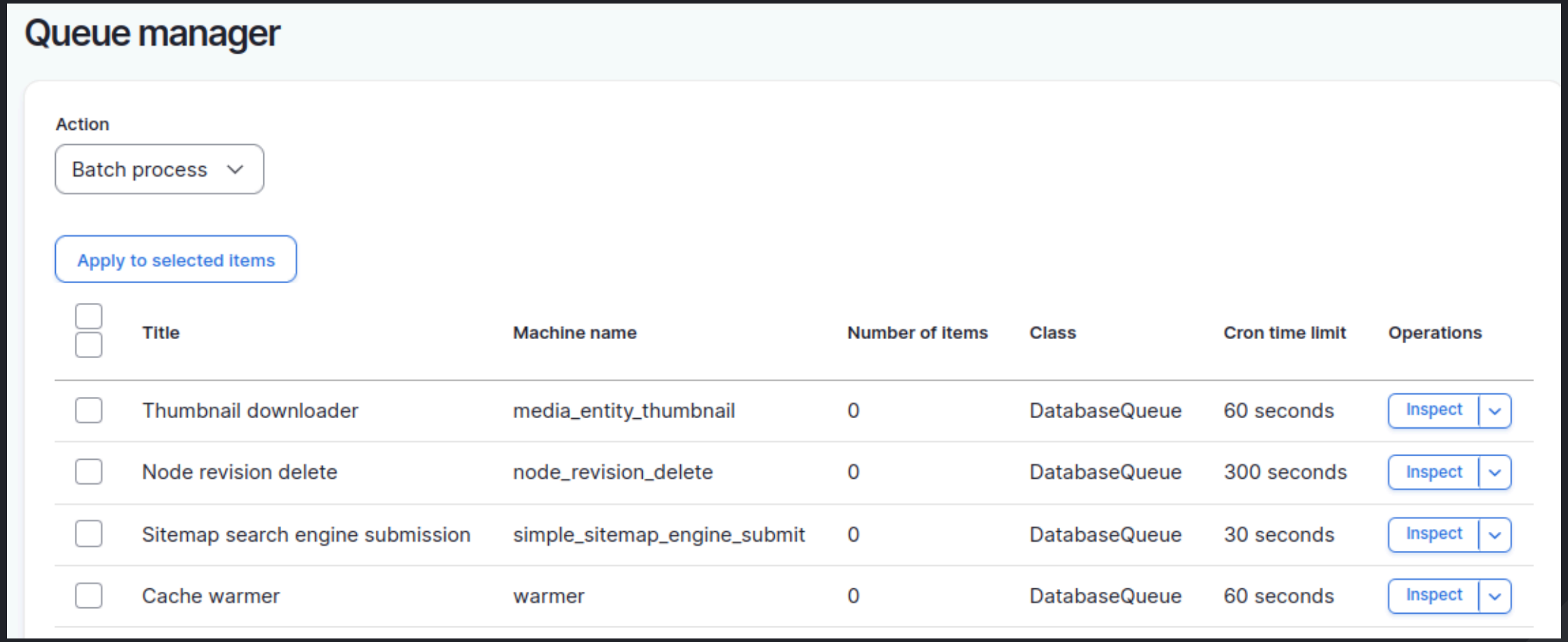
- There is no interface for queues so be sure to handle errors properly. Logging is important when things go wrong.
- If your users need to process something straight away then use a batch.

Modules That Use Queue

Queue UI

- View and run queues, also inspect their contents.

https://www.drupal.org/project/queue_ui



Warmer

- Warms the cache for content entities. Runs this process via a queue.
- The sitemap.xml processing is a good example of storing an array in the queue data.

https://www.drupal.org/project/warmer

Queue Unique

- Creates a custom queue storage handler to store all items in the queue as unique.

https://www.drupal.org/project/queue_unique

Resources

- Queues on `#! code`
  https://www.hashbangcode.com/tag/queues-api
- All code seen is code available at
  https://github.com/hashbangcode/drupal_queue_examples/
- Batch Processing and the Drupal Queue System -
  https://www.drupalatyourfingertips.com/bq

Questions?

- Slides: https://github.com/hashbangcode/queue-api-talk

Philip Norton hashbangcode.com fosstodon.org@hashbangcode
fosstodon.org@philipnorton42

Thanks!

- Slides: https://github.com/hashbangcod queue-api-talk

Philip Norton hashbangcode.com fosstodon.org@hashbangcode
fosstodon.org@philipnorton42