Laravel 事件及序列功能應用

by fripig

就是個看技術文章當興趣的肥宅
在癮科技當工程師
www.cool3c.com

github.com/fripig/article_log
www.plurk.com/fripig
twitter.com/fripig

先講序列(QUEUE)

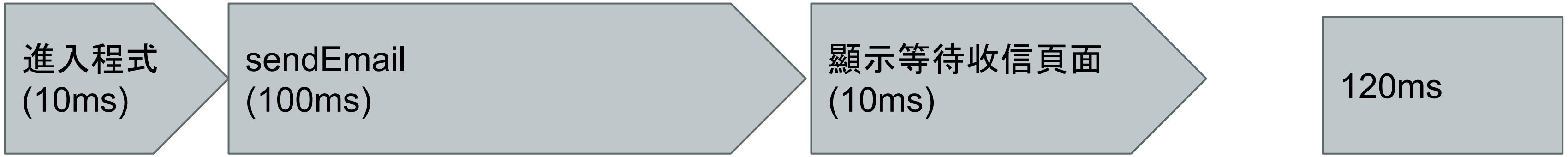
序列(QUEUE)

Queues allow you to defer the processing of a time consuming task, such as sending an email, until a later time. Deferring these time consuming tasks drastically speeds up web requests to your application.

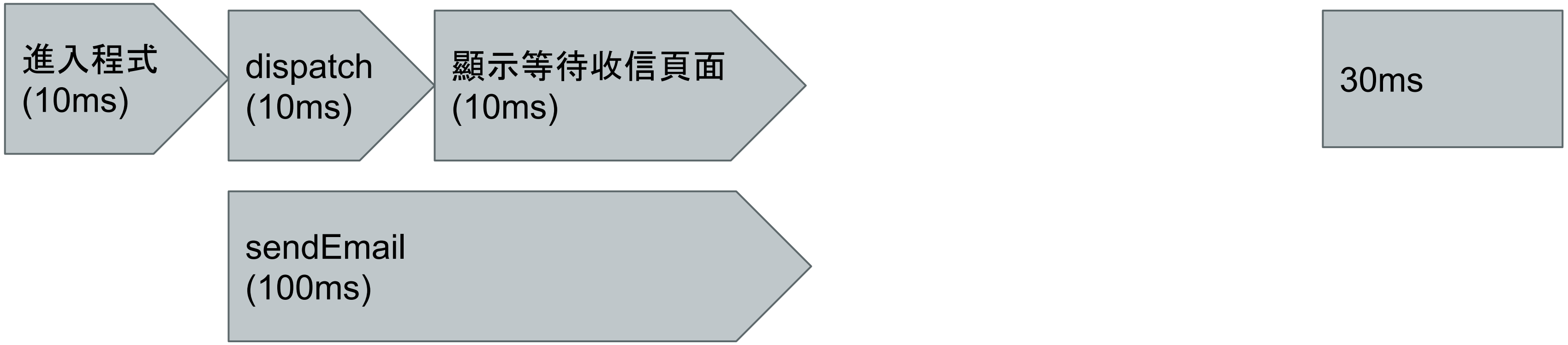
序列(QUEUE)
的優點

延時處理

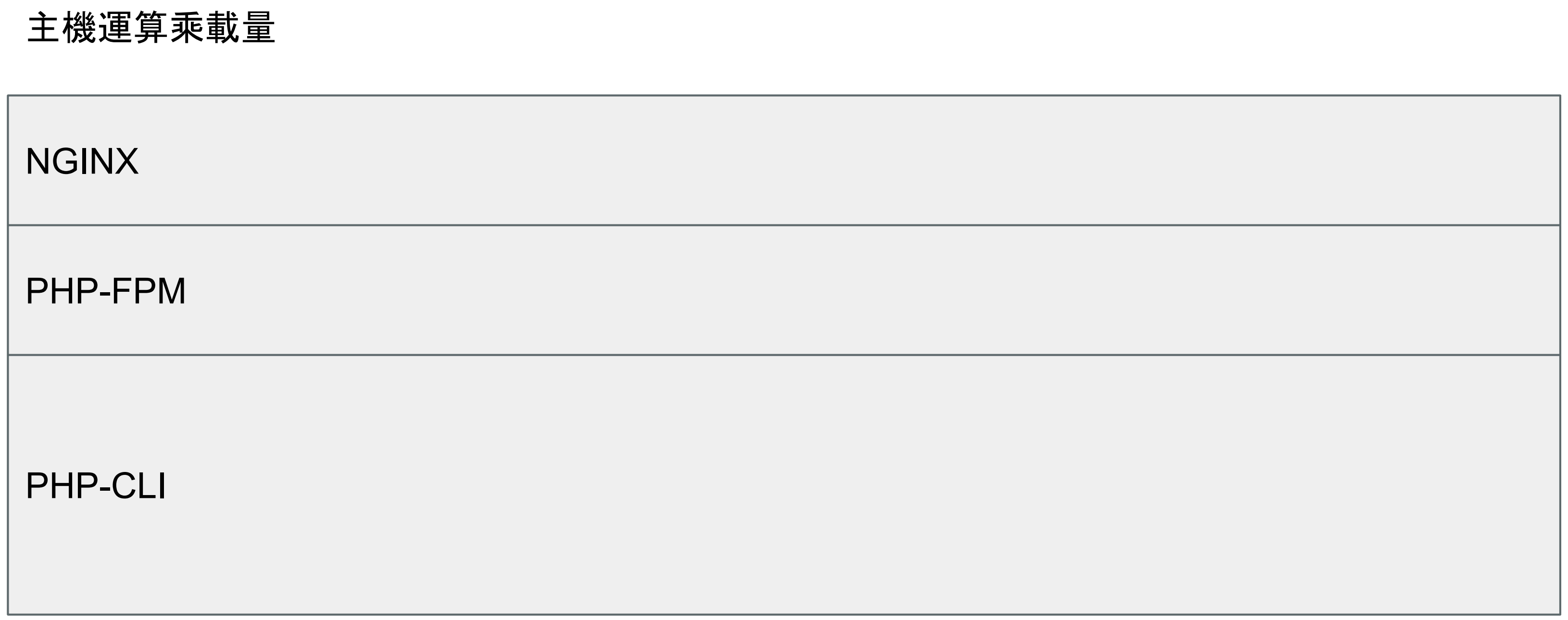
寄出認證信的運作順序



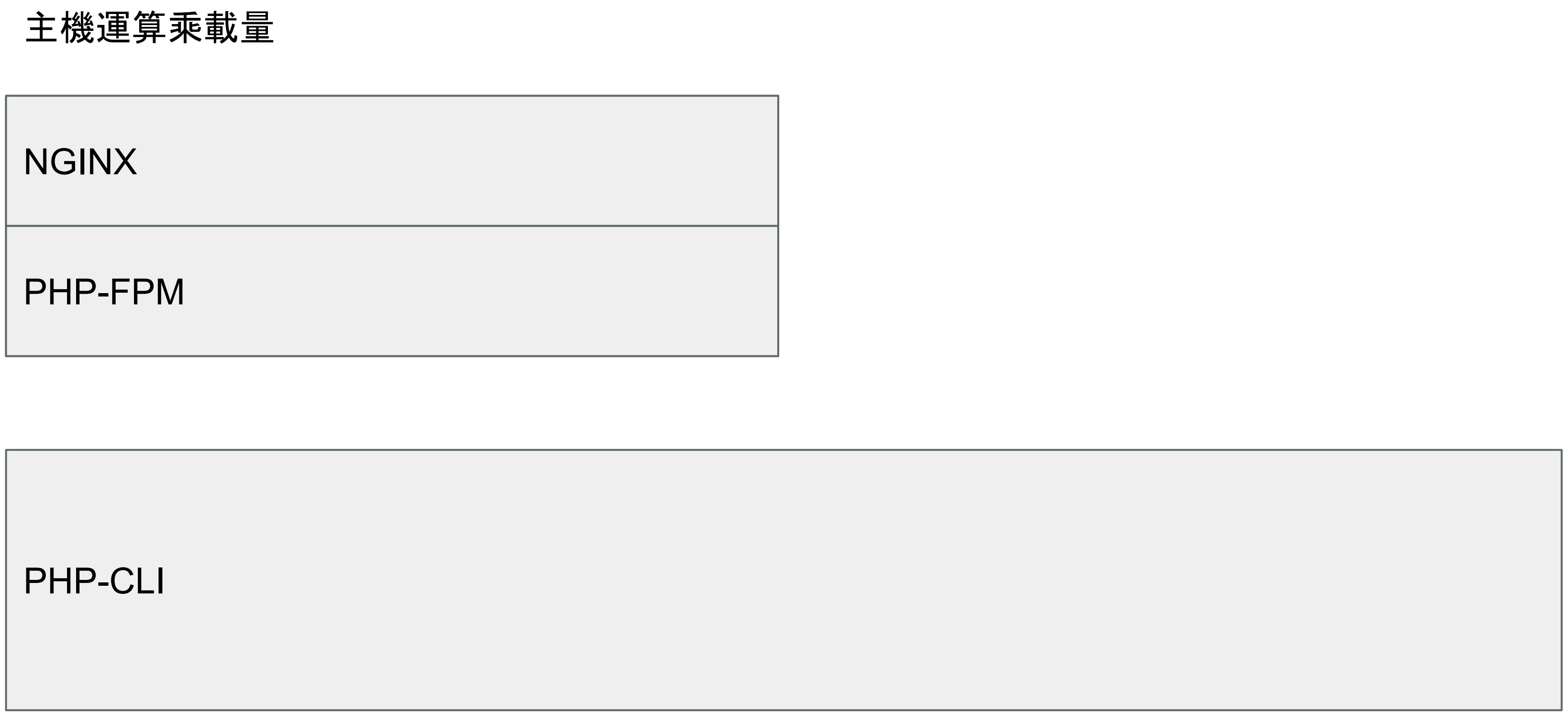
使用序列的運作順序



運算分離



沒有運算分離的系統設計



可以運算分離的系統設計

```
dispatch((new Job)->onConnection('highCPU'));
```

```
dispatch((new Job)->onQueue('high'));
```

```
php artisan queue:work highCPU --queue=high
```

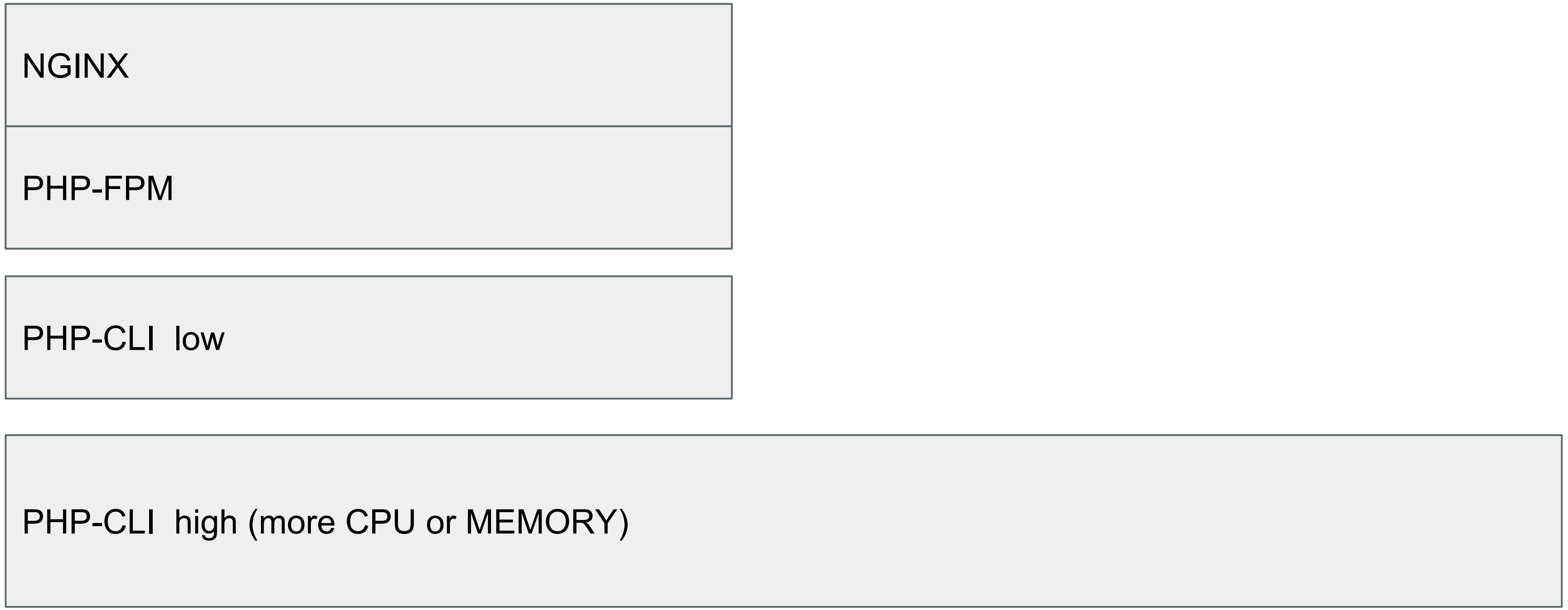
```
class PostCacheUpdate implements ShouldQueue
{
	public $queue = 'high';
}
```

```
php artisan queue:work --queue=high,default,low
```

```
php artisan queue:lis --queue=high,default,low
Processing: App\Listeners\PostCacheUpdate
PostCacheUpdate
Processed:  App\Listeners\PostCacheUpdate
Processing: App\Listeners\PostCDNUpdate
PostCDNUpdate
Processed:  App\Listeners\PostCDNUpdate
```

```
dispatch((new Job)->onQueue('high'))
```

主機運算乘載量



將序列工作分配到不同乘載量的電腦執行

降低耦合

```
php artisan make:job PublicPost
```

```php
namespace App\Jobs;

class PublicPost ProcessPodcast implements ShouldQueue

{

    use Dispatchable, InteractsWithQueue, Queueable, SerializesModels;

    protected $post;

    public function __construct(Post $post)

    {

        $this->post = $post;

    }

    public function handle()

    {

    }

}
```

SerializesModels

```
dispatch(new PublicPost($post));
```

要注意的是序列裡面

其實是用Model的ID去撈當時狀態的資料

而不是新增工作到序列時的資料

```
{
    "displayName": "App\\Jobs\\PublicPost",
    "job": "Illuminate\\Queue\\CallQueuedHandler@call",
    "maxTries": null,
    "timeout": null,
    "data": {
        "commandName": "App\\Jobs\\PublicPost",
        "command":
"O:19:\"App\\Jobs\\PublicPost\":5:{s:7:\"\u0000*\u0000post\";O:45:\"Illum
inate\\Contracts\\Database\\ModelIdentifier\":2:{s:5:\"class\";s:8:\"App\
\Post\";s:2:\"id\";i:1;}s:6:\"\u0000*\u0000job\";N;s:10:\"connection\";N;
s:5:\"queue\";N;s:5:\"delay\";N;}"
    }
}
```

可靠性

```
php artisan queue:work --timeout=30 --tries=3
```

```php
<?php

namespace App\Jobs;

class ProcessPodcast implements ShouldQueue

{

    public $tries = 5;


    public $timeout = 120;


   public function failed(Exception $exception)

    {

       // Send user notification of failure, etc...

    }


}
```

```
php artisan queue:failed-table

php artisan queue:failed

php artisan queue:retry 5

php artisan queue:retry all

php artisan queue:forget 5

php artisan queue:flush
```

- database,
- Beanstalkd
- Amazon SQS
- Redis
- sync
- null

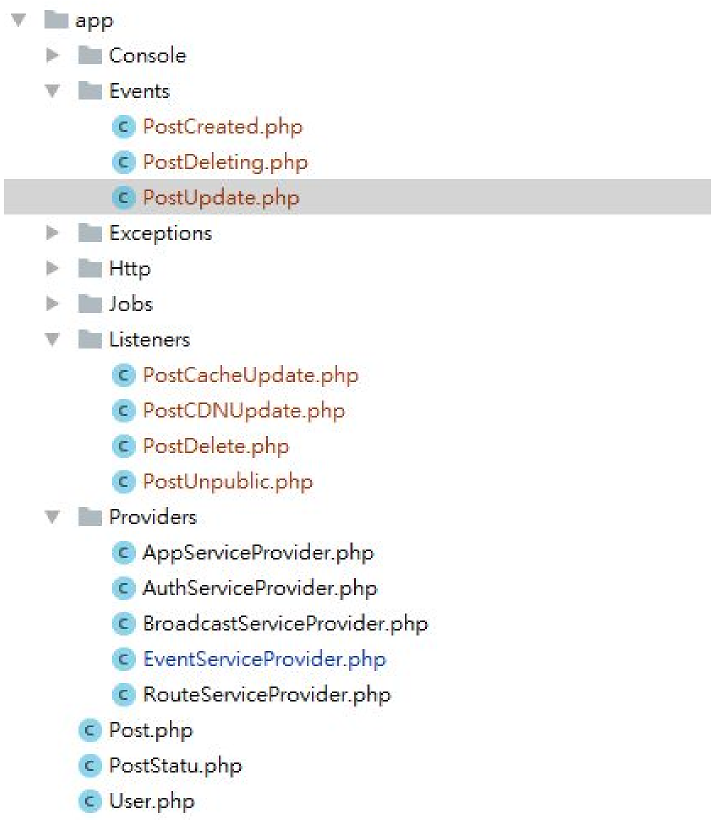
事件(EVENT)

Laravel's events provides a simple observer implementation, allowing you to subscribe and listen for various events that occur in your application.

EventServiceProvider.php

```php
namespace App\Providers;

class EventServiceProvider extends ServiceProvider
{
   protected $listen = [
       'App\Events\PostCreated' => [
           'App\Listeners\PostCDNUpdate',
       ],
       'App\Events\PostUpdate' => [
           'App\Listeners\PostCacheUpdate',
           'App\Listeners\PostCDNUpdate',
       ],
       'App\Events\PostDeleting' => [
           'App\Listeners\PostUnpublic',
           'App\Listeners\PostCacheUpdate',
           'App\Listeners\PostCDNUpdate',
           'App\Listeners\PostDelete',
       ],
   ];
}
```



```
app
  Console
  Events
    PostCreated.php
    PostDeleting.php
    PostUpdate.php
  Exceptions
  Http
  Jobs
  Listeners
    PostCacheUpdate.php
    PostCDNUpdate.php
    PostDelete.php
    PostUnpublic.php
  Providers
    AppServiceProvider.php
    AuthServiceProvider.php
    BroadcastServiceProvider.php
    EventServiceProvider.php
    RouteServiceProvider.php
  Post.php
  PostStatu.php
  User.php
```

```
php artisan event:generate
```

```php
<?php
namespace App\Listeners;
use App\Events\PostCreated;
class PostUnpublic
{
   public function __construct()
   {
       //
   }
   public function handle(PostCreated $event)
   {
       //
   }
}
```

```php
<?php
namespace App\Listeners;
use App\Events\PostEvent;
class PostUnpublic
{
   public function __construct()
   {
       //
   }
   public function handle(PostEvent $event)
   {
       //
   }
}
```

```
event(new PostUpdate($post));
```

```php
<?php

namespace App;
use Illuminate\Database\Eloquent\Model;

class Post extends Model
{
   protected $events = [
       'created' => \App\Events\PostCreated::class,
       'updated' => \App\Events\PostUpdate::class,
       'deleting' =>\App\Events\PostDeleting::class,
   ];
```

ShouldQueue

總結

https://divinglaravel.com/queue-system

DRY

Don't Repeat Yourself Principle

Q&A