Logging & Error Handling

PSR-3 Logger

- LoggerInterface
- Log levels
- Message with context placeholders
- Context can contain anything
- LoggerAwareInterface, LoggerAwareTrait

Monolog

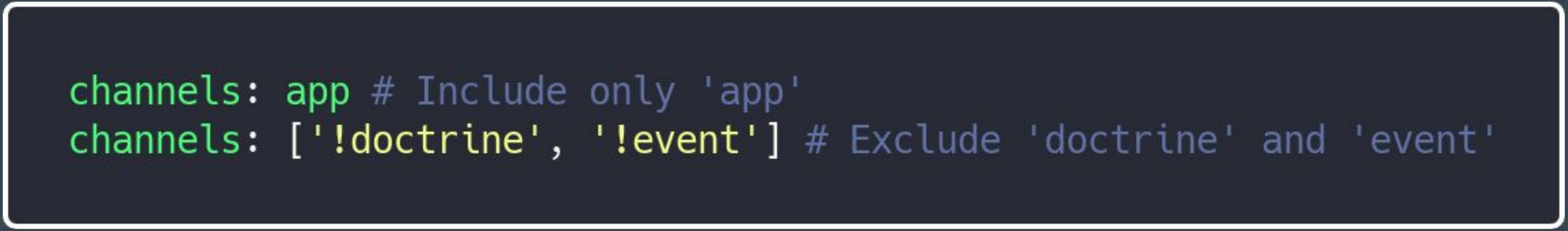
Very powerful

- Send alerts and emails
- Log to any external service or database
- PSR-3 support
- Wrappers
  - FingersCrossed
  - Deduplication
  - Filter
  - Etc.
- Extra data to log records
  - MemoryUsageProcessor

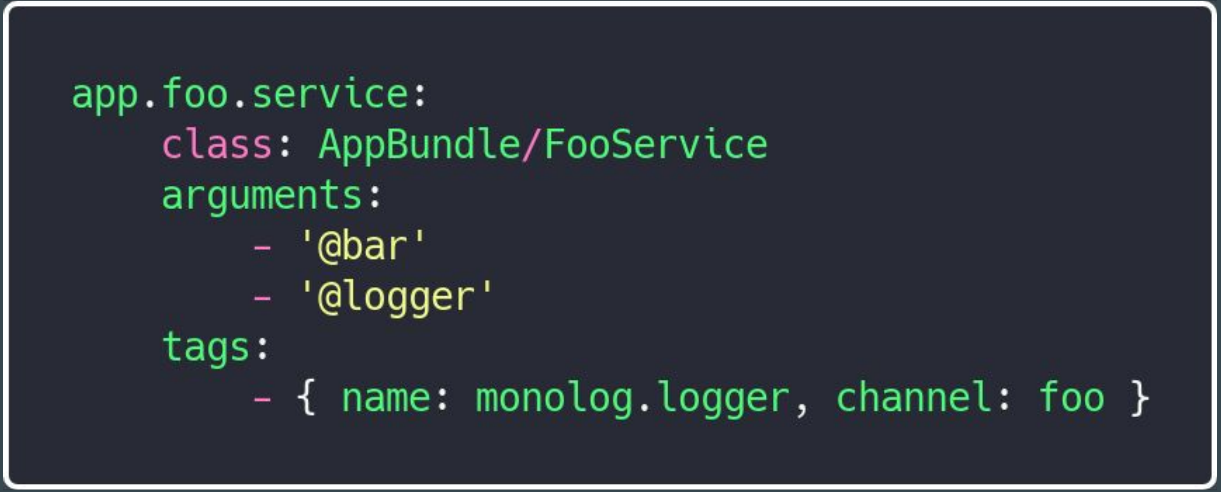
Logging Channels

A way to organize log messages

- tag all logger usages with channel name



```yaml
channels: app # Include only 'app'
channels: ['!doctrine', '!event'] # Exclude 'doctrine' and 'event'
```



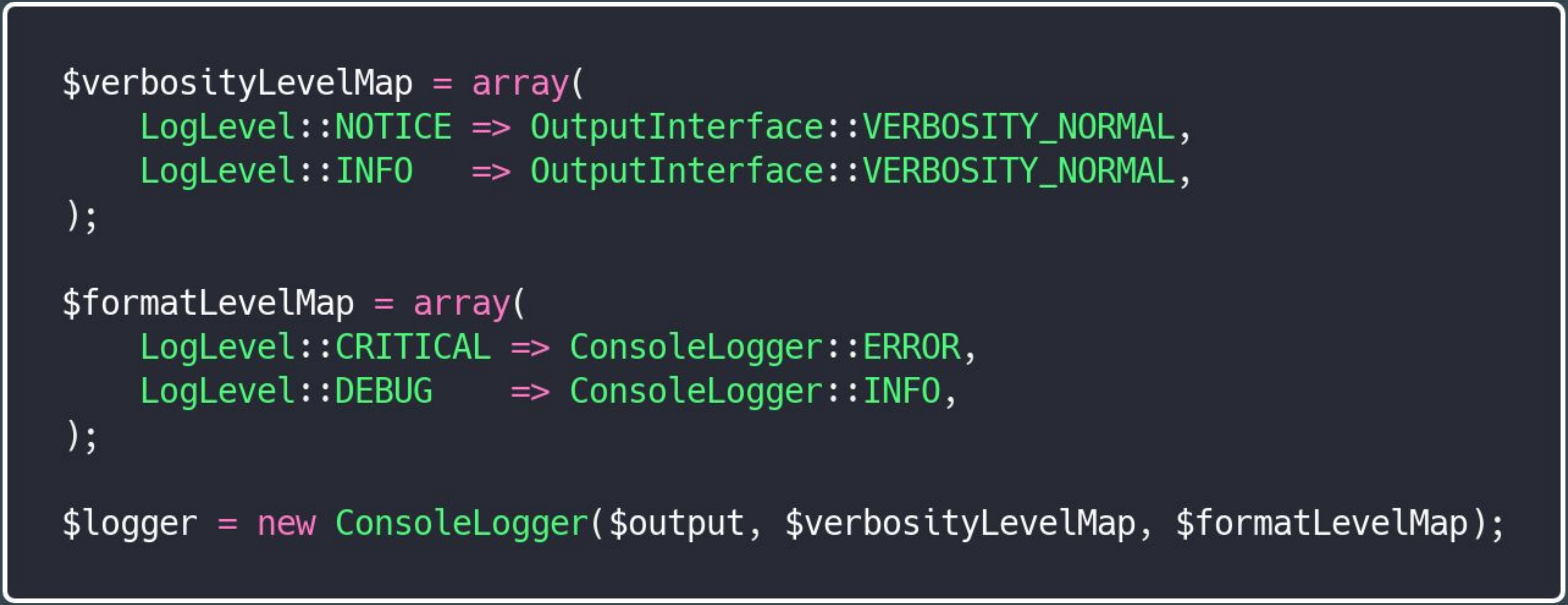
```yaml
app.foo.service:
    class: AppBundle/FooService
    arguments:
        - '@bar'
        - '@logger'
    tags:
        - { name: monolog.logger, channel: foo }
```

Configuration

- Error logs to emails
    - Config_prod.yml
    - LoggerBundle/README.md
- ORO monolog configuration

Console Integration

- ConsoleLogger
  - MQ integration example

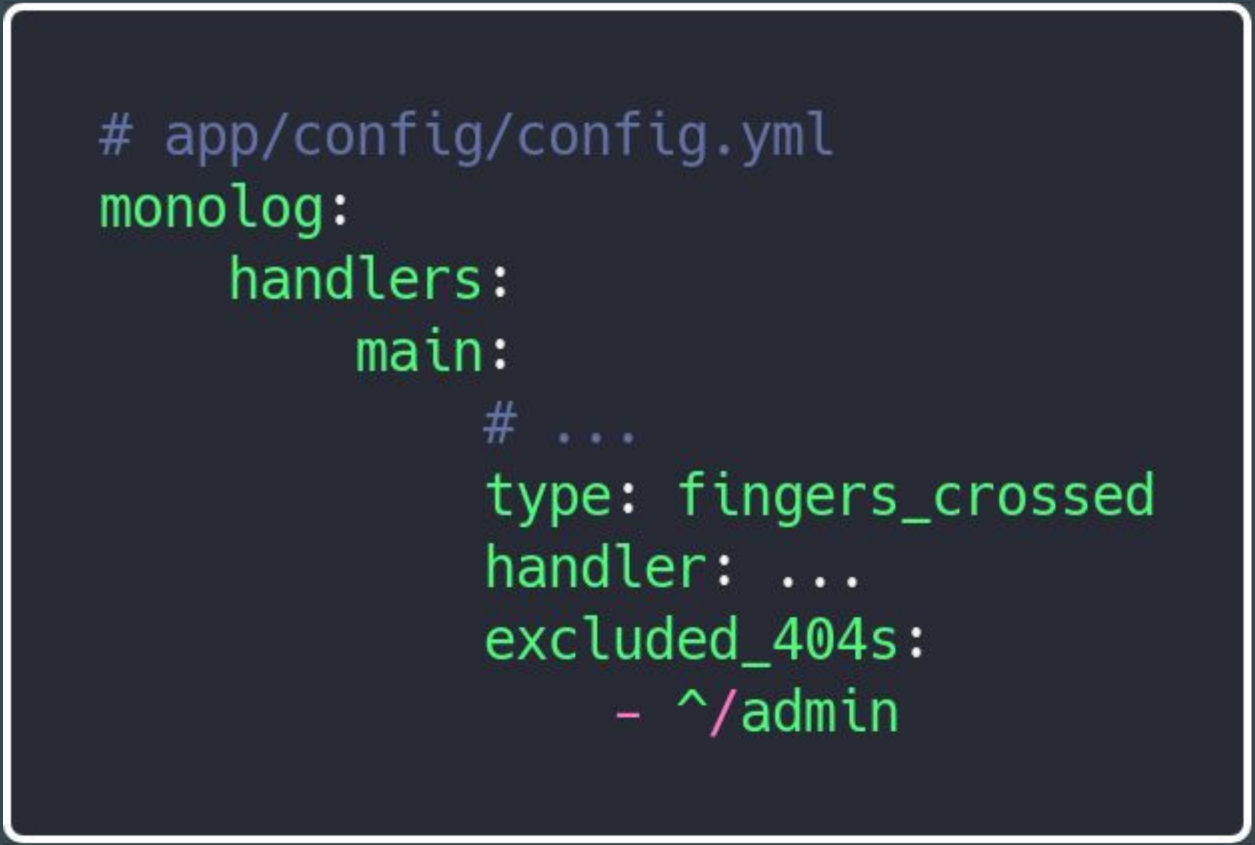


```
$verbosityLevelMap = array(
    LogLevel::NOTICE => OutputInterface::VERBOSITY_NORMAL,
    LogLevel::INFO   => OutputInterface::VERBOSITY_NORMAL,
);

$formatLevelMap = array(
    LogLevel::CRITICAL => ConsoleLogger::ERROR,
    LogLevel::DEBUG    => ConsoleLogger::INFO,
);

$logger = new ConsoleLogger($output, $verbosityLevelMap, $formatLevelMap);
```

Exclude 404 from PROD logs



```yaml
# app/config/config.yml
monolog:
    handlers:
        main:
            # ...
            type: fingers_crossed
            handler: ...
            excluded_404s:
                - ^/admin
```

Exception Handling

Good

```
catch (\Exception $e) {
    if (null !== $this->logger){
        $this
          ->logger
          ->error(
              $message,
              ['exception'=> $e]
            );
    }
    // optionally
    $this
      ->session
      ->getFlashBag()
      ->add('warning', $message);
    // recover
}
```

Bad

```
// empty catch
catch (Exception $e) {
}

catch (Exception $e) {
    // do some work without logging
    // or rethrowing
}

// catch language errors
catch (ErrorException $e) {
  // do some work
}
```

Custom Error pages

templates/bundles/TwigBundle/Exception/

- error{statusCode}.{format}.twig
- error403.html.twig
- error.html.twig
- error404.json.twig
- error403.json.twig
- error.json.twig

Tips

- Always tag service with injected @logger
- Don’t use sprintf
- Log background running console commands
  - At least errors (ORO do out-of-the-box)
- HTTP Aware exceptions only at Controllers
  - Symfony\Component\HttpKernel\Exception\NotFoundHttpException

Debugging

- Use proper log management service
  - Google Stackdriver
    - Example usage with MessageQueue
  - ELK
- Logger Configuration
  - config:dump-reference monolog
  - debug:config monolog
- debug_backtrace
- xDebug
- cat var/logs/prod.log | grep 'ERROR\|CRITICAL'