広告配信管理システム
を支えるPHP
レガシーシステムからの段階的移行戦略

2017/10/08 #phpcon2017

@suzu_v VOYAGE GROUP

私について

すずけん github: suzuken @suzu_v

fluct SSP https://fluct.jp を作っています

ajito.fm というポッドキャストをやっています

こんなケース、ありませんか？

原因を追えないログ「処理が正常に完了しませんでした」

年代によって書かれ方の異なるコードたち

なぜ動いているのかわからないが多分動いている

全体を把握できないくらい育ってしまった

そして、**これらのコードは価値がある（＝稼いでいる）**

https://upload.wikimedia.org/wikipedia/commons/6/6b/Jenga_distorted.jpg

価値あるコードを
より良くしたい

アジェンダ

👉背景、現状、規模

足固め: 例外基盤とモニタリング

改善: コードを消す、コーディングガイドライン

まとめ

現状の良い点悪い点

- 良い👍: 価値出してる・テストはある
- 悪い👎: 機能追加に時間がかかる・メンテナンスしづらい・読みづらい

規模

- モデル数: 120
- コントローラ数: 100
- テーブル数: 500
- 開発期間: 2009年末ごろから
- テスト: 350クラス。主にモデルと自社ライブラリ向け 、その他に主要機能の統合テストがある。

👍単体テスト、統合テストはすべてCIで自動実行される

👍コードレビューされなければmasterにマージしない。テストの無いコードはほぼマージされない。

👍リモートブランチごとに確認用管理画面が立ち上がる。確認しやすい。

それでも複雑になる…

下準備

- 静的解析で大まかに複雑そうなところを見つける:
  phpmd, phpcpd, phpdcd, PHP_CodeSniffier, phan etc.
- そしてgrepで重複コードを洗い出す

突然ですがこんなコードが300箇所ありました

```
public function index($id = null) {
    try {
        if ($id === null || $id === "") {
            throw new Exception(cmn_msg_param);
        }
        // 処理
    } catch (Exception $e) {
        log_message($this->cmn_const_message->getErrorStr($e->getCode()),
            $e->getMessage() . "(" $e->getFile().":".$e->getLine().")");
        $this->tpl->display('error.html',
            array("error" => $e->getMessage()));
    }
}
```

突然ですがこんなコードが300箇所ありました

```
public function index($id = null) {
    try {
        if ($id === null || $id =
            throw new Exception(cmn_msg_param);
        }
        // 処理
    } catch (Exception $e) {
        log_message($this->cmn_const_message->getErrorStr($e->getCode()),
            $e->getMessage() . "(" $e->getFile().":".$e->getLine().")");
        $this->tpl->display('e
            array("error" => $e->getMessage()));
    }
}
```

とりあえずルート例外をthrow

そしてルート例外でcatch

ログもエラー画面も同じメッセージ

突然ですがこんなコードが
200行ほどあったとします

画面出力

正常に動作を完了できませんでした。
操作が不正です

ログ

NOTICE 2017-10-04T00:12:08+09:00:
Exception: with message 操作が不正です
in /path/to/foo.php:104

ログもエラー画面も同じメッセージ

```
public function
    try {
        if ($id === null || $id === "") {
            throw new Exception(cmn_msg_param);
        }
        // 処
    } catch (
        log_m                                    ode()),
            $e->getMessage() . "(" $e->getFile().":".$e->getLine().")");
        $this->tpl->display('e
            array("error" => $e->getMessage()));
    }
}
```

突然ですがこんなコードが
500箇所ありました

画面出力

正常に動作を完了できませんでした。
操作が不正です

```
public function
    try {
        if ($id === null || $id === "") {
            throw new Exception(cmn_msg_param);
        }
        // 処
    } catch (
        log_m                                    ode()),
            $e->getMessage() . "(" $e->getFile().":".$e->getLine().")");
        $this->dp->display('
            array("error" => $e->getMessage()));
    }
}
```

ログ

NOTICE 2017-10-04T00:12:08+09:00:
Exception: with message 操作が不正です
in /path/to/foo.php:104

ログもエラー画面も同じメッセージ

何がうまくいかなかったのか
わからない => 怖い

再掲: こんなケース、ありませんか？

👉原因を追えないログ「処理が正常に完了しませんでした」

年代によって書かれ方の異なるコードたち

👉なぜ動いているのかわからないが多分動いている

全体を把握できないくらい育ってしまった

そして、**これらのコードは価値がある（＝稼いでいる）**

アジェンダ

背景、現状、規模

👉足固め: 例外基盤とモニタリング

改善: コードを消す、コーディングガイドライン

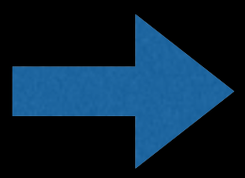
まとめ

例外とハンドリング

1. アプリケーションベース例外を定義する
2. アプリケーショングローバルな例外ハンドラを実装 http://php.net/set_exception_handler
3. ロギングとエラー表示をいい感じにする

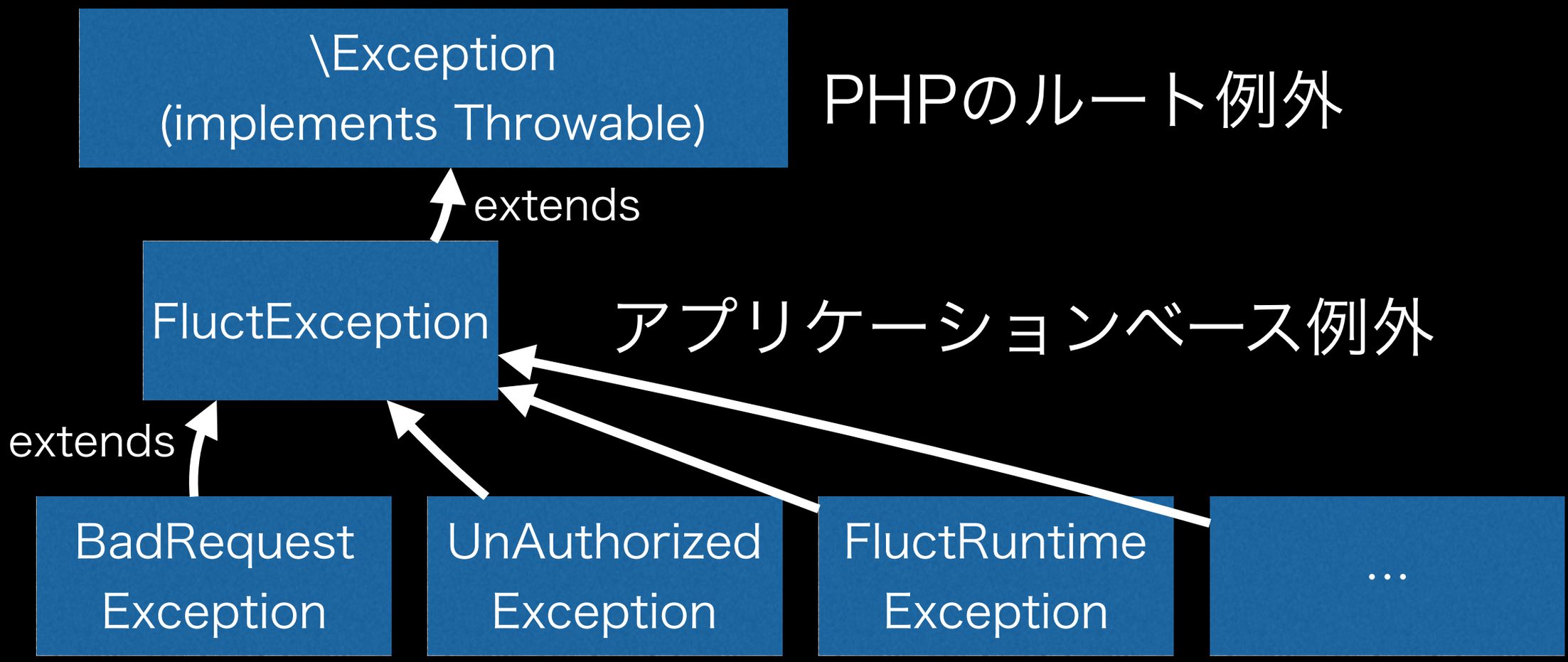
set_exception_handler

```php
<?php
set_exception_handler("handler");

function handler($ex)
{
    printf("%s 🚀", $ex->getMessage());
}

throw new Exception("例外は怖くない");
```



例外は怖くない 🚀

アプリケーションベース例外階層



* PHPの標準例外はアプリケーションベース例外にいれない
  * FluctThrowableインタフェースにして標準例外もキャッチできるようにしようか？と考えたがやめた
* 結果FluctExceptionでcatchすればよし。シンプル。

アプリケーションベース例外

ルート例外

ユーザ向けに
表示するメッセージ

```php
<?php
namespace Fluct\Exception;

class FluctException extends \Exception implements Throwable
{
    protected $userMessage;

    public function __construct($message = "", $code = 0,
        \Exception $previous = null, $userMessage = "")
    {
        parent::__construct($message, $code, $previous);
        $this->userMessage = $userMessage;
    }

    public function getUserMessage()
    {
        return $this->userMessage;
    }
}
```

```php
class BadRequestException extends FluctException
{
    public function __construct($message="BadRequest", $code=400,
        \Exception $previous=null, $userMessage="入力が不正です。")
    {
        parent::__construct($message, $code, $previous, $userMessage);
    }
}
```

```
+      * @throws BadRequestException
       */
      public function detail($id = null)
      {

@@ -71,11 +75,7 @@ public function detail($id = null)
          if (! $data) {
-             // エラーログ出力
-             log_message('NOTICE', 'It is an illegal request');
-             $this->smarty_parser->parse('ci:cmn/error.tpl', array());
-
-             return;
+             throw new BadRequestException();
          }
```

例外導入前: その場しのぎのロギングとエラーページ表示

例外導入後: 👍例外をthrowすればOK

例外ハンドラ

```php
$ci =& get_instance();
$container = $ci->pimple->container;
$twig = $container['twig'];
if ($e instanceof FluctException) {
    if ($e->getCode() !== 0) {
        header('HTTP', true, $e->getCode());
    } else {
        header('HTTP', true, 500);
    }
    $error_message = $e->getUserMessage();
    // NewRelic側で例外ログをだしているのでここではNOTICEレベルとする
    log_message("NOTICE", get_class($e) . ": with message "
    $e->getMessage() . " in " . $e->getFi
    "\n" . $e->getTraceAsString());
    $twig->display('cmn/error.twig', compact('error_message'));
    return;
}
```

例外コードによって
HTTPレスポンスコードを変える

スタックトレースも
ログに追加

エラー表示もいい感じにやる
（テンプレートはコンテナから差し込む）

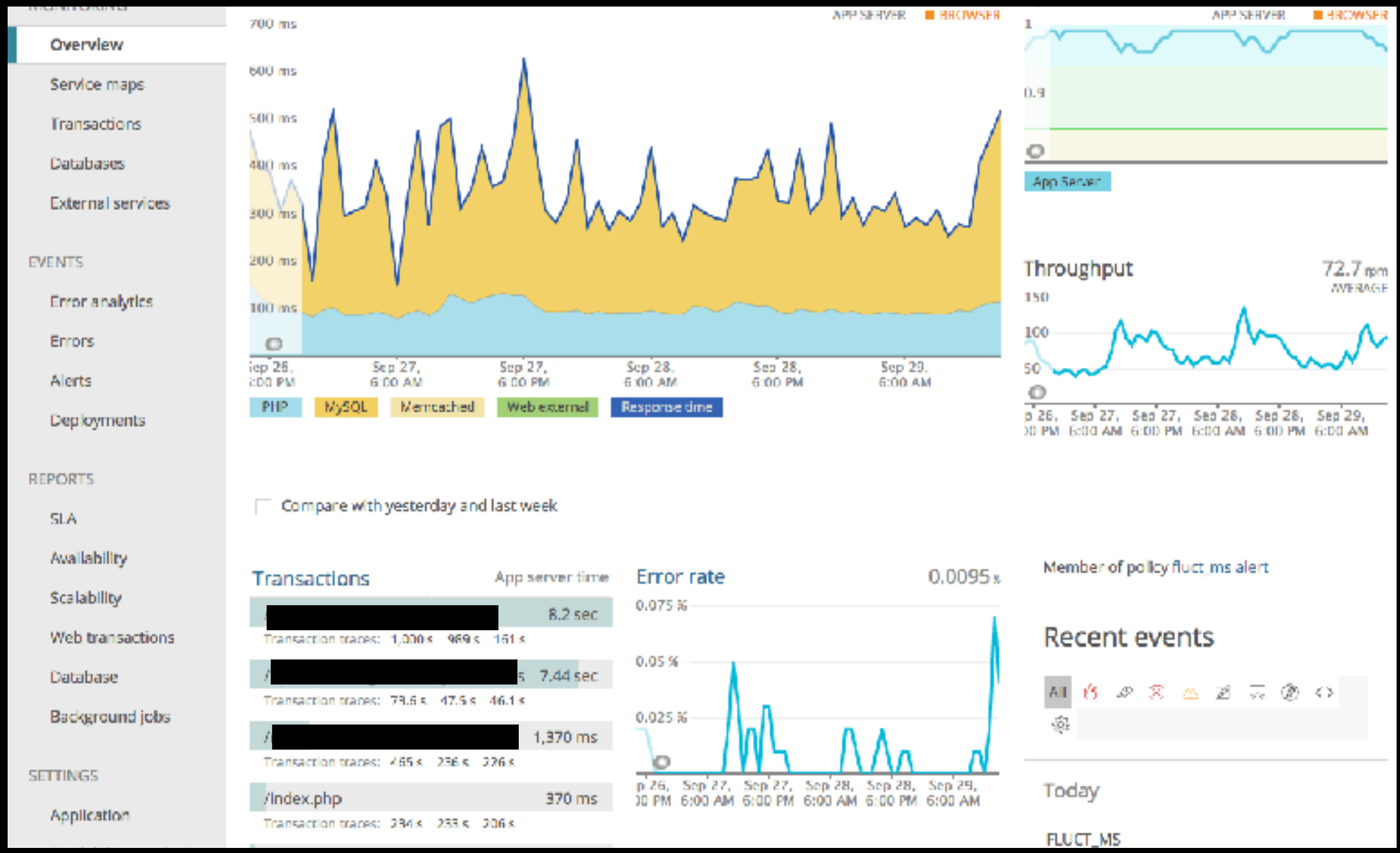
remove catch(Exception $e) on controllers #2909

Merged **suzuken** merged 1 commit into `master` from `t99999_throw_exceptions` on 16 Nov 2016

Conversation 5 Commits 1 Files changed 24

Changes from all commits ▾ Jump to... ▾ +43 −1,051

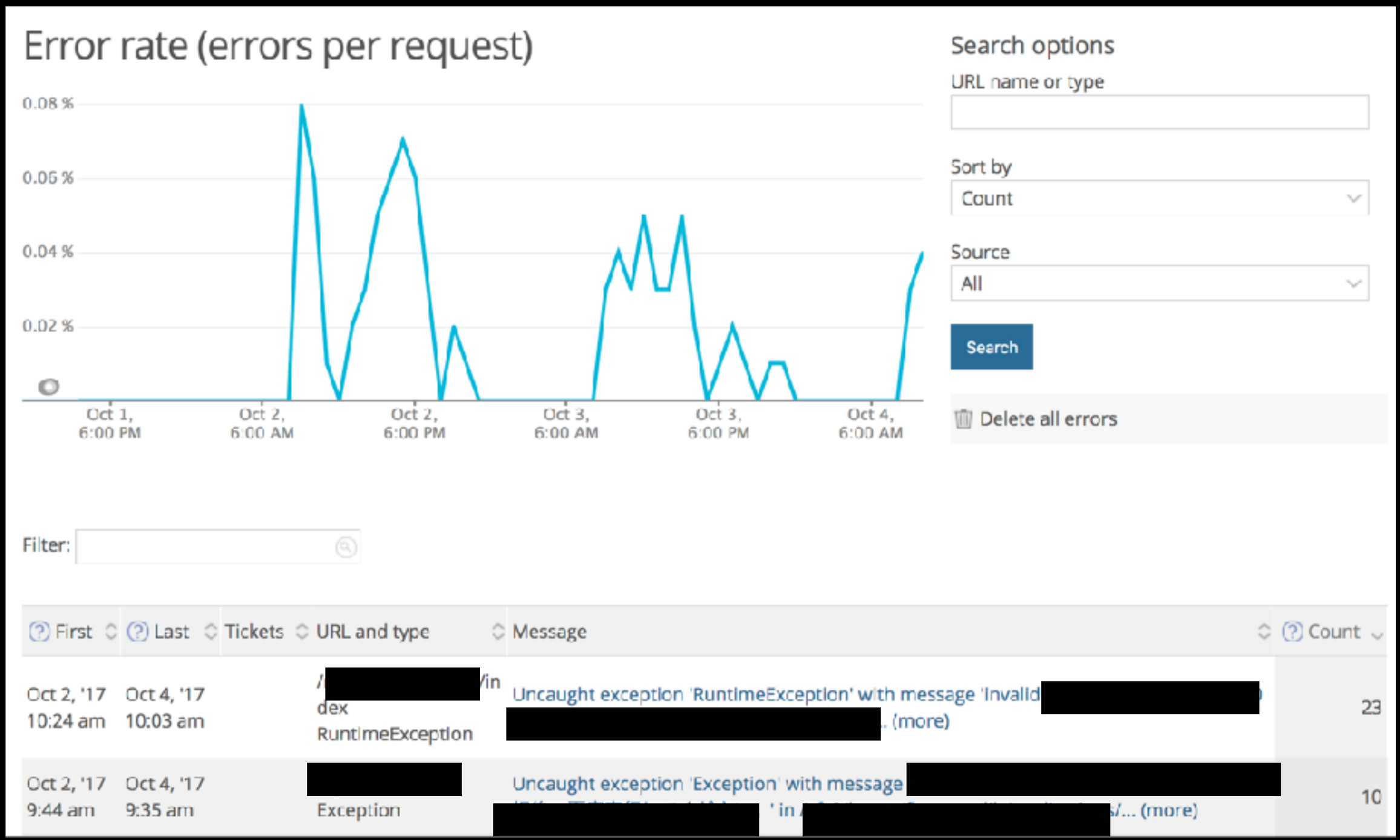
エラーページ出力とロギングが共通化されたので
try catchの重複コードを安全に削除可能に 👍



モニタリング

NewRelic APM

NewRelic Errors



モニタリング: NewRelic採用

👍例外基盤をつくったことで、どの画面でいつどんな例外がどれくらい発生したかわかりやすくなった

👍NewRelicアラートに含まれる情報が例外をベースにしており、問題の切り分けがしやすくなった

👍遅いクエリ、遅い画面もわかるようになり「ここをみれば必要な情報は集まっている」状態になった

アジェンダ

背景、現状、規模

足固め: 例外基盤とモニタリング

👉改善: コードを消す、コーディングガイドライン

まとめ

再掲: こんなケース、ありませんか？

原因を追えないログ「処理が正常に完了しませんでした」

👉年代によって書かれ方の異なるコードたち

なぜ動いているのかわからないが多分動いている

👉全体を把握できないくらい育ってしまった

そして、**これらのコードは価値がある（＝稼いでいる）**

PhpStorm

- **PHPだけど、PHPじゃなくなります**
- ものすごい強いInspection機能。@property や @var で挙動を壊さず型をつけられます。
- dead code削除やrefactoringに大活躍。チームで使ってます。

宣伝👂: https://ajito.fm/9/

PHP 5.3 -> 5.6 -> …

- phpcs + php7ccで洗い出しつつ基本はPHPマニュアルを読んで粛々と直す
- なにか問題あればNewRelicでわかる
- フレームワークのコアもPHP 5.6向けに修正

```
$ phpcs --standard=PHPCompatibility lib
```

参考: http://enzolutions.com/articles/2015/06/07/how-to-check-php-compatibility/

どんどん消す💪

- Google Analyticsで使ってない画面を探る + 担当者に聴いて確かめる。 -> 使ってないならまるっと**消す**
- unused codeをひたすら地道に**消す**（PhpStormのinspection）
- 要らないテストケースを**消す**（遅いテストは生産性を落とす）

Redashに頼り、消す

- 社内でもSQLかける人が増えてきた
- 「それRedashでよくない？このクエリでさくっとデータ出せますよ」
- 利用頻度の少ない画面はRedashにクエリだけ移行して担当者に案内 + まるっと削除

最良のコードは、コードなし

https://blog.codinghorror.com/the-best-code-is-no-code-at-all/

“It's painful for most software developers to acknowledge this, because they love code so much, but **the best code is no code at all**.”

- NewRelic, Google Analytics, Redash etc.
- 外部ツールにうまく頼ろう。いらないコードはどんどん消そう。

実装ガイドラインを作る

- PSR-1,2 準拠 + php-cs-fixerのCI組み込み
- クラス名関数名などの命名ルールを決める
- PHPDoc @param @return を推奨。動的なクラスローダーは @property @var で補助。
- @deprecatedをつかおう

などなど

地道に消す

Deprecations

非推奨な関数、クラス、プロパティ等には以下のように `@deprecated` アノテーションを追加すること。

```
/**
 * @deprecated この関数は非推奨になりました。代わりにSome()をつかってください。
 */
```

また、 `E_USER_DEPRECATED` エラーをトリガーすることで、ロギング及びエラーの収集を可能にします。以下のようにします。

```
@trigger_error("this sample() is deprecated in 2016-10-11", E_USER_DEPRECATE);
```

クラスの場合には `__construct()` の中で宣言するとよいでしょう。

Proposal: コードを消すプロセスについて も参考にしてください。

fluctコーディングガイドラインより抜粋

そしてペアプログラミング

- ガイドラインを整えたら、ペアプログラミング
- ペアプログラミングで書き方を伝える（レビューだとぶつかりやすいときに特に有用）
- でも、強制してはいけない

アジェンダ

背景、現状、規模

足固め: 例外基盤とモニタリング

改善: コードを消す、コーディングガイドライン

👉まとめ

価値あるコードを
より良くしたい

誰だって不安はある

「この変更で挙動を壊したらどうしよう」

だから、守りをまず固める。
そうすれば改善しやすくなる。

コードを

自分たちの手に取り戻す