パッケージを作ろう！ LT

Create your PHP library/package!

pixiv

2017-05-22 PHP BLT #7

お前誰よ

- うさみけんた（@tadsan）/ Zonu.EXE
  - GitHub/Packagistでは id:zonuexe
- ピクシブ株式会社でひたすらPHPやってる p(ixi)v
- フレームワークなき道にフレームを作るおしごと
- 最近やってること
  - ひたすらに要らないコードを整理したり、
    コアの部分をぐにゃっとできるように悩んでるよ
  - Mastodon API Client/SDK作ったよ

pixiv

Mastodon

たのしいんですよ

PHPからMastodonのAPIを簡単に触れるクラスを作った

```
$composer require
zonuexe/mastodon-api
```

コンポーザーを手元に入れる方法はキータに書いた

パッケージ作るときに決めること

- パッケージ名（vendor/package）
- 対象PHPバージョン
- ディレクトリ構成
- ライセンス

pixiv

パッケージ名

- パッケージ名はvendor/package 形式（[-0-9a-z]+ -と数字小文字のみ）
- GitHubと一致させておくと親切
- zonuexe/mastodon-api にした

pixiv

対象PHPバージョン

- どこから対応するか？ HHVMは？
  - 5.3.2+（Composer最低）
  - 5.5+（旧安定版、サポート終了済み）
  - 7.0+（旧安定版）
  - 5.6+, 7.1+（現安定版）

pixiv

ディレクトリ構成

- 拘りがなければsrc/とtests/に置く
  `git init; mkdir src/ tests/;`

- コマンドを配布したければ `bin/` に

- `docs/` にドキュメントを置くと、
  GitHub Pagesに展開できて便利

pixiv

オートローディング

- モダンなPHPにrequireは最低限
- 各クラスにincludeとか書かない
- composer.jsonに設定を書く
- 起動スクリプトでだけ
  autoload.phpを読み込むようにする

pixiv

オートローディング(PSR-4)

```
{
    "autoload": {
        "psr-4": {
            "Mastodon\\": "src/"
        }
    },
    autoload-dev": {
        "psr-4": {
            "Mastodon\\": "tests/"
        }
    }
}
```

pixiv

オートローディング(files)

- クラスは遅延ロードできるが、
  関数はできないので読み込む

```
{
    "autoload": {
        "files": ["src/functions.php"]
    },
    "autoload-dev": {
        "files": ["tests/helpers.php"]
    }
}
```

pixiv

オートローディング(classmap)

- 名前空間とディレクトリ構成が
  一致しない雑なPHPも力強く読める

```
{
    "autoload": {
        "classmap": ["src/"]
    },
    "autoload-dev": {
        "classmap": ["tests/"]
    }
}
```

pixiv

ライセンス（利用許諾）

- 二次利用させるためのガイドライン
- 好き勝手に利用させたいなら
  Apache-2.0かMITがオススメ
- 他人が行なった変更を合法的に
  自分にも取り込みたいならGPL-3.0

pixiv

極論

- ライセンス(利用許諾)を明示せずにソースコードをパブリックな場所に置いてはいけない
- そのソースコードを見付けたひとが合法的に再利用することができない

pixiv

よしこれで
準備が整った

パッケージを探す

- 定番パッケージはawesome-php
  https://github.com/ziadoz/awesome-php
- Packagistでぐぐる
  https://packagist.org/

pixiv

鉄則

- composer.jsonのrequireは弄るな
- composer requireはコマンドで。
  JSONファイルはいい感じに変更される
- バージョンはいい感じに解決される

```
$ composer require guzzlehttp/guzzle respect/validation
$ composer require --dev phpunit/phpunit:^4.8
```

pixiv

SDKがやること

SDKのおしごと

- HTTPリクエストする
- それをPHPにマッピングする
- 複雑さを隠すための工夫をする

pixiv

Entity

- Mastodon本家のAPIリファレンスをひたすらひたすらひたすら写してく
  https://github.com/tootsuite/documentation/blob/master/Using-the-API/API.md#entities
- 自作のライブラリで型付け
  https://github.com/BaguettePHP/objectsystem

pixiv

あんまり複雑なことを見せない

- ちょーかんたんっぽく

```
<?php use Baguette\Mastodon as m;
$mastodon = m\session(
    'pawoo.net', $client_id, $client_secret, [
        'scope' => 'read write follow',
        'grant' => ['username' => $mail, 'password' => $passwd],
]);

$account = $mastodon->getAccount(42);
$mastodon->postStatus(m\toot('トゥートゥー'));
```

pixiv

OOPであることには拘らない

- Requestするメソッド

```
public static function getAccount(
    Client $client, SessionStorage $session, $id
) {
    v::intVal()->min(0)->assert($id);
    return static::map(
        Entity\Account::class,
        $client->requestAPI('GET',
        sprintf('/api/v1/accounts/%d', $id), [], $session)
    );
}
```

pixiv

中では再帰的にクラスを生成

- APIの結果をデコードしてぶっこむ

```
public function __construct(array $properties) {
    $this->setProperties(mapValues($properties, [
        'account'                => Account::class,
        'reblog'                 => Status::class,
        'created_at'             => \DateTimeImmutable::class,
        'media_attachments' => [Attachment::class],
        'mentions'               => [Mention::class],
        'tags'                   => [Tag::class],
        'application'            => Application::class,
    ]));
}
```

pixiv

テストの書きかた

テスト

- がんばりすぎない
- 踏み外したくない道をしっかり書く
- HTTPリクエストはGuzzleで隠蔽

pixiv

Mastodon

たのしいんですよ

まだ未実装なAPIは
たくさん／(^o^)＼