

# Propelをあき

# らめる前に～ソースを

# 読むことでわかる限界突破法～

**kunit**

TAKAHASHI Kunitiko

Maple Project

株式会社ディノ

第42回PHP勉強会@関東; 2009-04-26(日);

# kunit

✓ 高橋邦彦

✓ 株式会社デイン

✓ 今月末まで

# 本日のお品書き

- ✓ **symfonyとは**
- ✓ **Propelの使い方**
- ✓ **Propelの限界突破法**

# 今日のまとめ

- ✓ Propelはまだまだ使える
- ✓ ソースを読もう

# symfonyとは

- ✓ Mojavi改 +  $\alpha$ 
  - ✓ Agaviとは兄弟
  - ✓ Mojaviを使っていた人はすんなり使える
- ✓ Mojaviの良いところも悪いところも引き継いでる
  - ✓ 1.0 → 1.1 → 1.2とバージョンが上がるたびに影響は薄まってる

# Mojavi

- ✓ 1クラス1アクション
  - ✓ Rails以前(=Struts型)のフレームワークの典型
- ✓ ディレクトリ構造が深い
  - ✓ 細かく設定を分けることができる
- ✓ MVCフレームワーク
  - ✓ Mの部分の支援はなし

# symfony

- ✓ 1クラスnアクション
  - ✓ 1クラス1アクションで使うこともできる
- ✓ Mojavi改+ $\alpha$ 
  - ✓ + $\alpha$ の部分で設定ファイルの形式のばらつきがある
    - ✓ ini/yaml/XML
    - ✓ 当初に比べるとyamlに統合されつつある

# フルスタック

- ✓ MojaviにはなかったMの支援
  - ✓ 当初はPropelのみだった
  - ✓ 今はDoctrineも選べる
  - ✓ とういなかDoctrine優勢？
    - ✓ jobeetへのリンクが . . .
    - ✓ Doctrineのことはfivestarに聞け

# Propel

- ✓ O/Rマッパー
  - ✓ JavaのTorqueがベースになってる
- ✓ ファイル生成を多用する
  - ✓ 設定ファイルからクラスを結構たくさん作る
- ✓ 今日はsymfony 1.0系でやります
  - ✓ 仕事で使っていたのが1.0だったので
  - ✓ 1.1/1.2ではPropelのバージョンがあがってる
    - ✓ さよならCreole . . .

# ライフサイクル

- ✓ 設定ファイルを書く
  - ✓ schema.yml
- ✓ テーブルを生成する
  - ✓ schema.ymlから生成できる
- ✓ モデルを生成する
  - ✓ lib/model以下にファイルがたくさんできる
- ✓ モデルを編集する
  - ✓ lib/model以下のファイルを編集する

# ファイル構成

- ✓ lib/model直下のファイルは自由に編集できる
- ✓ lib/model/om以下には継承元ファイルが生成されている
  - ✓ 再度propel-build-modelをしてもこちらが再生成される
  - ✓ つまり自分が編集したファイルは上書きされない
  - ✓ バージョン管理にはこのomディレクトリやmapディレクトリは入れない方がいい

# Criteria

- ✓ 絞り込み条件やソート条件等を指定する
- ✓ テーブルのフィールド名等はクラスの定数になってる
  - ✓ コード補完がないとつらい
  - ✓ 逆にコード補完ができると気持ちいい

# 使ってみよう

- ✓ テスト/学生/テスト結果というテーブル
- ✓ テスト一覧
- ✓ 学生一覧

# 今日の本題

- ✓ 簡単なものはCriteriaでOK
- ✓ けどね
  - ✓ 80:20の法則
  - ✓ 20にはいったときが・・・

# たとえば

- ✓ 各テストでの各教科の平均点の集計
- ✓ 各教科がある点数以上のものだけ絞り込むという検索
  - ✓ 例のための例なので無理矢理です
  - ✓ 次のようなSQLでできる

```
SELECT
  id,
  name,
  japanese,
  math,
  english
FROM
(
  SELECT
    MAX(test.id) AS id,
    MAX(test.name) AS name,
    AVG(japanese) AS japanese,
    AVG(math) AS math,
    AVG(english) AS english
  FROM
    test
    JOIN
      test_result
      ON test_result.test_id = test.id
  GROUP BY
    test.id
) AS test_list_view
WHERE
  japanese >= 70
```

# SQLの実行

- ✓ Propelレベルではなく一つ下のレベルになる
  - ✓ 1.0系だとCreoleレベル
  - ✓ 1.1以降だとPDOレベル
- ✓ Modelにはならない
  - ✓ sfPropelPagerが使えない

# さあ、困った

- ✓ ソースを読んでいる
- ✓ doSelectのコード
  - ✓ doSelectRSをpopulateObject  
している
  - ✓ doSelectRSの結果はRecordSet
  - ✓ Creoleを直接使ったのと同じ
  - ✓ なんとかなるんじゃない？

# さあ、どうする

- ✓ `schema.yml`
- ✓ 先ほどまでの例
  - ✓ 設定がテーブルと1対1対応
- ✓ けどそうじゃないといけないわけではない
- ✓ 次のような設定ができる

```
test_list_view:
```

```
  id:
```

```
  name:      { type: varchar(64) }
```

```
  japanese:  { type: integer, default: 0 }
```

```
  math:      { type: integer, default: 0 }
```

```
  english:   { type: integer, default: 0 }
```

```
  _attributes:
```

```
    readOnly: true
```

```
    skipSql: true
```

# 呪縛からの解放

- ✓ テーブルと1対1ではないモデル
- ✓ 自分で書いたSQLからPropel Modelをつくる
- ✓ sfPropelPagerが使える
  - ✓ sfPropelPagerはModelのdoSelectとdoCountを使ってるだけ

# 解放されたけど

- ✓ SQLを自分で組み立てる必要がある
  - ✓ SQLインジェクション対策を自分でするの？
- ✓ Criteriaにセットすれば条件ができるのは捨てがたい
- ✓ その部分だけができるものがあればいい
  - ✓ sfPropelQueryBuilderというのを作った
  - ✓ PropelのBasePeerをみればすぐに作れる

# 今日のまとめ

- ✓ Propelはまだまだ使える
- ✓ ソースを読もう

ご静聴

ありがとうございます

ございました