さわって実感

リレーショナルデータベース設計

7

アプリケーション設計（2）

日本PostgreSQLユーザー会 理事長 石井達夫 ISHII Tatsuo
ishii@postgresql.jp

PostgreSQLコンファレンス2002

6月15日都内で「PostgreSQLコンファレンス2002 in Tokyo」が開催されました．日本PostgreSQLユーザー会（JPUG）単独で技術コンファレンスを開くというのが私の長年の夢の一つでした．JPUG結成（1999年）から3年目にしてこの目的をようやく達成できたわけです．コンファレンスは90人ほどの定員の会場を2つ使ってパラレルにセッションを流す方式で開催され，入場料が無料であることもあり，定員を大幅に超える事前申し込みをいただきました．おかげで急遽セミナー受講券を作成するなど，会場整理に手間がかかりましたが，幸い事故もなくコンファレンスを終了することができました．

コンファレンスでは合計11個のセッションが開催されましたが，どれも盛況で，中には立ち見が出るものもありました．プログラム内容は，RDB入門のチュートリアルから，独自にPostgreSQLにレプリケーションを実装した報告まで，どれも非常に濃い内容でした．また，PHPユーザー会の大垣靖男さんによるセッションをはじめ，PHPに関連したセッションも複数あり，PostgreSQLとPHPの縁の深さを実感した次第です．

コンファレンス終了後はJPUGの年次総会が開催され，私自身ももう1年理事長の任を仰せつかりました．

そして一番最後に急遽パネルディスカッションが行われ，なんとPHPユーザー会の廣川類さん，そして本誌の連載でおなじみの小山哲志さんを含めた豪華メンバでパネルとなりました．このお二人は前号で紹介した通称青マンモス本（『PHP4徹底攻略 実戦編』）の著者でもあり，お二人のトークを同時に聞けた出席者は実にラッキーだと言えましょう．

Apache，PHPはアップデートラッシュ

前回はApache 1.3.24を使いましたが，Apacheにセキュリティホールが見つかり，1.3.26にアップデートされています．できれば1.3.26を使ったほうがよいでしょう．

PHPのほうは，前回は4.1.2でしたが，その後4.2.2がリリースされています．こちらもさまざまなバグフィックスのほか，マルチバイト対応の正規表現関数が正式に組み込まれているので，可能ならばアップデートしたほうがよいでしょう．インストール方法は前回の記事とまったく同じです．ただし，前回の記事の207ページの図5でphpinfo()関数の実行結果でPHPロゴの画像が壊れているのが正常，と説明しましたが，PHP 4.2.2ではこのロゴは正常に表示されます（図1）．

ちなみに，筆者のプラットフォームもVine Linux 2.5CRに更新されました．それにともなってブラウザがNetscape 4からMozilla 0.98になりました．まとめると，今号の記事を執筆するにあたり，次の環境で検証を行っています．

7 アプリケーション設計（2）

- Vine Linux 2.5CR（kernel 2.4.18で起動）
- PostgreSQL 7.2.1
- Apache 1.3.26
- PHP 4.2.2
- Mozilla 0.9.8

ソースコードの構成

それでは前回約束した通り，スクリプトの解説を行います．スクリプトが比較的大きいため，今回と次回の2回に分けて解説を加えます．なお，今回バグ修正のために表1のファイルを修正しています．お手数ですが，ご利用前に最新版をダウンロードしてください（http://www.gihyo.co.jp/ wdpress/）．

図2をご覧ください．トップレベルにはメインスクリプトであるindex.phpがあり，どの画面もこのスクリプトが表示処理のメインプログラムの役割を果たします．アプリケーション各機能はすべてlibディレクトリの下にあるファイルに分割してあり，それぞれのファイルには「モデル」クラスと「ビュー」クラスが記述されています．これらのクラスはコントローラクラスである“Controller”からincludeされます．一方，PHPLIBのクラスライブラリはすべてphplibサブディレクトリの下に置いてあります．

トップレベルのディレクトリ下には以下のような.htaccessファイルが置いてあり，PHPスクリプトをinclude/requireするパスの設定などが記述されています．

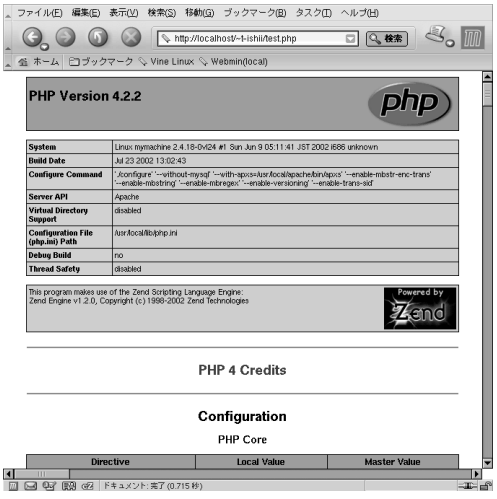
```
# includeパスの設定
<IfModule mod_php4.c>
php_value include_path ".:./lib:./phplib"
</IfModule>

# "magic quote"をオフにする
<IfModule mod_php4.c>
php_flag magic_quotes_gpc off
</IfModule>
<IfModule mod_php4.c>
php_flag magic_quotes_runtime off
</IfModule>
```

サブディレクトリには以下のような.htaccessが置いてあり，Web経由では参照できないようになっています．

```
Order deny,allow
Deny from all
```

図1　PHP 4.2.1のphpinfoの画面



メインスクリプト

それでは，メインスクリプト（index.php，リスト1）の説明に入ります．

セッション管理開始宣言

リスト1-①はPHPLIBを使う上での決まり文句のよ

▼表1　修正したファイルの一覧

ファイル名	最新レビジョン番号
index.php	1.3
ib/articles.inc	1.3
lib/iss_mags.inc	1.2
lib/magazines.inc	1.2
lib/model.inc	1.2

うなものです．webdb_prepend.incはPHPLIBに含まれるprepend.php3をコピー，変更したもので，このファイルの中で基本的なPHPLIBのクラスライブラリをロードします．webdb_prepend.incの中ではwebdb_local.incをロードします．webdb_local.incは，使用するデータベースなどの指定を行います．

page_open文は，PHPLIBによるセッション管理の開始を宣言する関数です．引数はセッション管理などに使用するクラス名の指定です．ここでは，webdb_local.incで定義したExample_Session（セッション管理クラス）とExample_Auth（認証クラス）を指定しています．

page_openでセッションの開始を宣言したら，必ずページの最後にpage_closeを呼び出さなければなりません（リスト1-⑥）[注1]．これを忘れるとセッション管理のデータが更新されませんので，注意してください．

セッション管理対象データの宣言

PHPLIBでは，セッション管理の対象となる変数はセッション管理クラスのメンバ関数registerを使います（リスト1-②）．

ログアウト処理

URL引数で“logout”が渡ってきた場合は，ログアウト処理を行います．具体的には，認証クラスのlogout関数を呼び出します（リスト1-③）．

コントローラクラスのロード

セッション管理変数の“class”がセットされている場合は，controller.incをロードし，コントローラに制御を渡します（リスト1-④）．

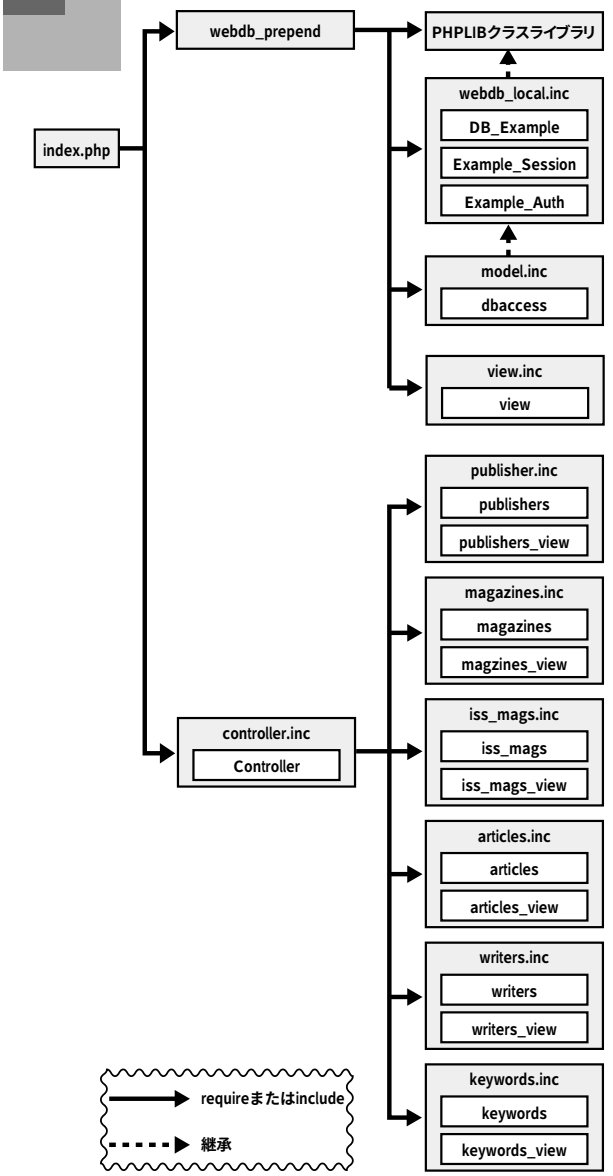
操作対象データベースの選択処理

セッション管理変数の“class”がセットされていない場合は，操作対象データベースの選択を行うためのメニューを表示します（リスト1-⑤）．

コントローラクラス

コントローラクラス（リスト2）は，MVCモデル（前号参照）の中で制御を担当する部分です．このクラスでは，次の5つのセッション変数を使います．

図2 ソースコードの構成



注1）PHPLIBドキュメントの和訳がhttp://www.php.gr.jp/php/phplib/で参照できます．詳細はこちらをどうぞ．

7 アプリケーション設計（2）

original_query　：検索を実行したときのSELECT文
offset　：ページ位置（0スタート）
defvals　：表示中のデータ内容
def　：フォームに表示されている値
ctid　：現在表示中の行のTID

モデルとビューのクラスをロード

最初の仕事は，処理するテーブルに合わせて適切なモデルとビューのクラスをロードすることです．「モデル」はMVCモデルにおいては業務ロジックや業務オブジェクトの状態を管理し，「ビュー」は結果の見え方を司るものでしたね．

今回のプログラムでは，モデルはセッション変数“class”に格納されている文字列がそのままモデルクラスのクラス名になります．ビューのほうは，

「モデルクラスのクラス名」＋「_view」

という命名規則にしてあります．また，モデルとビューは「モデル」.incというファイルに定義されています（リスト2-①）．

アクションと対応する処理の実行

ユーザからのアクションはGETあるいはPOSTメソッドによって入力されてきます．アクションに対する操作は原則としてモデルクラスによらずすべて決まっており，同一のメンバ関数を呼び出して処理を実行します（リスト2-②）．

●アクションが「update」の場合

更新ボタンが押された状態です．exec_updateを呼び出してDBの更新処理を行い，exec_searchで更新後の結果を再表示するデータを作ります．

リスト1　メインスクリプト（index.php）

```
<?php
require_once("webdb_prepend.inc");
$class="";
page_open(array("sess"=>"Example_Session",
        "auth"=>"Example_Auth"));                    ①

$sess->register("class");  ②
$url = strip_tags($sess->self_url());

print<<<EOF
<html>
<head><title>雑誌データベース管理システム
</title></head>
<body bgcolor="white">
<h1>雑誌データベース管理システム</h1>
EOF;

if (isset($_REQUEST["logout"])) {
  $auth->logout();  ③
  print<<<EOF
ログアウトしました。
再度ログインする場合は<a href="index.php">
ここ</a>をクリックしてください。
EOF;
  page_close();
  exit;
} else if (isset($_REQUEST["class"])) {
  $class = $_REQUEST["class"];
} else if (isset($_REQUEST["return"])) {
  $class = "";
}

if ($class != "") {
  include_once("controller.inc");  ④
} else {
  print <<<EOF
処理対象のデータを選択してください．

<form method="POST" action="$url">
<select name="class">
<option value="publishers">出版社
<option value="magazines">雑誌
<option value="iss_mags">発刊雑誌
<option value="articles">記事
<option value="writers">著者
<option value="keywords">キーワード
</select>                                            ⑤
<input type="submit" value="選択">
<hr>
<input type="submit" name="logout" value="ログアウト">
</form>\n
EOF;
}
print <<<EOF
</body>
</html>
EOF;
page_close();  ⑥
?>
```

●アクションが「insert」の場合

追加ボタンが押された状態です．exec_insertを呼び出してDBの追加処理を行い，exec_searchで更新後の結果を再表示するデータを作ります．

●アクションが「delete」の場合

削除ボタンが押された状態です．exec_deleteを呼び出してDBの削除処理を行い，exec_searchで更新後の結果を再表示するデータを作ります．

リスト2　コントローラクラス（Controller.inc）

```
<?php
require_once("webdb_prepend.inc");

$sess->register("original_query");
$sess->register("offset");
$sess->register("defvals");
$sess->register("def");
$sess->register("ctid");

// modelとviewのクラスをロード
include_once("$class.inc");
$model =& new $class();
$view_class = $class . "_" . "view";
$view =& new $view_class();                                    ①

if (isset($_REQUEST["update"])) {
  // 「更新ボタン」
  if (is_array($defvals)) {
    $model->exec_update($ctid, $defvals,
      $_REQUEST["val"]);
    $defvals = $model->exec_search(FALSE,
      $offset, $_REQUEST["val"]);
  }

} else if (isset($_REQUEST["insert"])) {
  // 「追加」ボタン
  if ($model->exec_insert($defvals,
    $_REQUEST["val"])) {
    $offset = 0;
    $defvals = $model->exec_search(TRUE,
      $offset, $def, $_REQUEST["val"]);
  }

} else if (isset($_REQUEST["delete"])) {
  // 「削除」ボタン
  if ($model->exec_delete($ctid, $defvals)) {
    $defvals = $model->exec_search(FALSE,
      $offset, $def);
  }

} else if (isset($_REQUEST["first_page"])) {
  // 「検索開始」ボタン
  $offset = 0;
  $defvals = $model->exec_search(TRUE, $offset,
    $def, $_REQUEST["val"]);
} else if (isset($_REQUEST["new_first_page"])) {
  // 「新規追加検索開始」ボタン
  $offset = 0;
  $defvals = $model->exec_new_search(TRUE,
    $offset, $def, $_REQUEST["val"]);

} else if (isset($_REQUEST["next_page"])) {
  // 「次ページ」ボタン
  if ($offset != "") {
    $offset++;
    $defvals = $model->exec_search(FALSE,
      $offset, $def);
  }

} else if (isset($_REQUEST["previous_page"])) {
  // 「前ページ」ボタン
  if ($offset != "") {
    if ($offset != "" && $offset > 0) {
      $offset--;
    }
    $defvals = $model->exec_search(FALSE,
      $offset, $def);
  }

} else if (isset($_REQUEST["cand_reg"])) {
  // 執筆者あるいはキーワードの「追加」ボタン
  $col = key($_REQUEST["cand_reg"]);
  $model->exec_candidate_reg($col,
    $_REQUEST["candidates"][$col], $defvals);
  $defvals = $model->exec_search(FALSE, $offset,
    $def);

} else if (isset($_REQUEST["cand_del"])) {
  // 執筆者あるいはキーワードの「削除」ボタン
  $col = key($_REQUEST["cand_del"]);
  $model->exec_candidate_del($col,
    $_REQUEST["candidates"][$col], $defvals);
  $defvals = $model->exec_search(FALSE, $offset,
    $def);

} else {
  $original_query = "";
  $offset = "";
  $defvals = "";
  $def = "";
  $ctid = "";
}

$def = $view->search_form($model, $defvals);                   ③

page_close();
?>
```

●アクションが「first_page」の場合

検索開始ボタンが押された状態です．exec_searchで検索を行います．

●アクションが「new_first_page」の場合

新規追加検索開始ボタンが押された状態です．exec_new_searchで検索を行います．

●アクションが「next_page」の場合

次ページボタンが押された状態です．offsetを変更してからexec_searchで次ページの検索を行います．

●アクションが「previous_page」の場合

前ページボタンが押された状態です．offsetを変更してからexec_searchで前ページの検索を行います．

●アクションが「cand_reg」の場合

執筆者あるいはキーワードの「追加」ボタンが押された状態です．記事テーブルだけの特殊な処理で，exec_candidate_regを呼び出してwritingsあるいはarticles_keywordテーブルに追加処理を行います．

●アクションが「cand_del」の場合

執筆者あるいはキーワードの「削除」ボタンが押された状態です．記事テーブルだけの特殊な処理で，exec_candidate_delを呼び出してwritingsあるいはarticles_keywordテーブルから削除処理を行います．

ビューの更新処理

以上でビューを更新する準備ができたので，ビュークラスのsearch_formを呼び出して表示画面を更新します（リスト2-③）．

テーブルごとのモデルクラスとビュークラス

コントローラクラスは単にユーザからのボタン入力に応じて必要なメンバ関数を呼び出すだけで，実際に何が起きているかを知るためには，テーブルごとに用意されているモデルとビューのクラスを見ていく必要があります．また，ほとんどの処理は，テーブルごとのクラスが継承しているdbaccessクラスおよびviewクラスに記述されています．各テーブルごとのクラスは実際にはテーブルごとの固有のパラメータを設定して

リスト3　magazinesクラス（magazines.inc）

```
<?php
require_once("publishers.inc");

class magazines extends dbaccess
{
  function cols() {
    return array("mag_name" => TRUE,
      "list_price" => TRUE);
  }                                          ①

  function magazines() {
    dbaccess::dbaccess();                    ②
    $this->read_query =<<<EOF
      SELECT p.pb_name, m.mag_name,
      m.list_price, m.publisherid,
      m.magazineid, m.ctid
      FROM publishers AS p,
        magazines AS m
      WHERE p.publisherid
        = m.publisherid
EOF;                                         ③
    $this->new_reg_query =<<<EOF
      SELECT p.pb_name, '' AS mag_name,
        '' AS list_price,
      p.publisherid, '' AS magazineid,
        '' AS ctid
      FROM publishers AS p
EOF;                                         ④
    $this->write_table = "magazines";        ⑤
    $this->write_cols = $this->cols();       ⑥
    $this->key = array("magazineid");        ⑦
    $this->foreign_key =
      array("publisherid" => TRUE);          ⑧
    $this->parent_tables
      = array("publishers");
  }
}

class magazines_view extends publishers_view
{
  function magazines_view() {
    publishers_view::publishers_view();      ⑨
    $this->show_cols =
      array_merge($this->show_cols,          ⑩
        magazines::cols());
  }
}
?>
```

いるだけです．このような構造にしておくことにより，テーブルの構造が変わったり，新しいテーブルを追加しても容易に対応できるようになります．

各テーブルごとのクラスは，記事テーブルを除いて非常によく似ています．全部を解説するのは無駄ですので，本稿では「雑誌」テーブルと「記事」テーブルの関連クラスを解説します．

雑誌テーブル関連のクラス

雑誌テーブル関連のクラスは，lib/magazines.incに格納されています（リスト3）．このファイルには，magazinesクラスとmagazines_viewクラスが格納されています．

magazinesクラス

本質的な処理はすべてdbaccessクラスから継承したメンバ関数の中で行います．実際にDBにアクセスする処理は後でdbaccessクラスを説明するときに解説します．ここでは，テーブルごとのパラメータについて説明します．

●cols関数

この関数（リスト3-①）はこのテーブルが持つ列名の連想配列を返します．連想配列のキーを列名，配列要素の値はPHPで論理値の「真」を表す定数である“TRUE”を設定します．この関数が返す列は表示されたデータの中で，このテーブルに所属する列を示します．

●コンストラクタ

PHPでは，クラス名と同じ名前のメンバ関数がコンストラクタとなります．コンストラクタの役割は以下となります．

- **親クラスのコンストラクタの呼び出し（リスト3-②）**

PHPでは，継承先で自動的に親クラスのコンストラクタが呼び出されません．そこで明示的に親クラスのコンストラクタを呼び出します．ここでは，データベースへの接続などの処理が行われます．

- **read_query変数の設定（リスト3-③）**

この変数には検索を行うときのSELECT文をセットします．実際には，このSELECT文にさらに“AND ....”のようにしてユーザが入力した検索条件を追加します．この例のように，親テーブルであるpublishersテーブルを結合させるSELECT文を書くこともできます．検索結果の列には，必ず以下を含んでおかなければなりません．

・cols関数の返す列（m.mag_name，m.list_price）
・主キー（m.publisherid，m.magazineid）
・本テーブルのTID（m.tid）

- **new_reg_query変数の設定（リスト3-④）**

このテーブルの新しい行を追加する際に，親テーブルであるpublishersテーブルを検索するためのSELECT文を設定します．read_queryとよく似たSELECT文になりますが，ポイントは，publishersテーブル以外の検索結果の列はダミーとして空文字列‘ ’を設定しておくことです．

- **write_table変数の設定（リスト3-⑤）**

更新対象のテーブル名を設定します．

- **write_cols変数の設定（リスト3-⑥）**

更新対象となる列名を設定します．

- **key変数の設定（リスト3-⑦）**

主キーとなる列名を設定します．

- **foreign_key変数の設定（リスト3-⑧）**

外部キーとなる列名を設定します．

magazines_viewクラス

こちらは簡単で，単にコンストラクタを設定するだけです．magazinesクラスと同様に親クラス（publishers_view）のコンストラクタを呼び出します（リスト3-⑨）．show_cols変数には実際に表示される列名を設定します（リスト3-⑩）．ここでは，親クラスの列にmagazinesクラスの列を追加する設定を行っています．

viewクラス

それでは，実際に処理をしている中核のクラスの一つであるviewクラスを説明します．

リスト4　viewクラス（view.inc）

```
<?php
$col_names = array (
  "pb_name" => "出版社名",
  : (省略)
  );
// view クラス
class view
{
  var $show_cols;// 表示対象列名リスト

  // フォーム表示
  function search_form($dbmodel, $def = "") {
    global $col_names;

    $meta = $dbmodel->get_meta(); ①

    print <<<EOF
      <form method="POST"> ②
      <table>\n
EOF;
    foreach ($meta as $ar) {
      $col = $ar["name"];
      if (isset($this->show_cols[$col])) {
  $name = $col_names[$col];
  $val = "val[$col]";
  if ($def != "" && isset($def[$col])) {
    $defval = $def[$col];
  } else {
    $defval = "";
  }
  if ($dbmodel->is_writable_col($col)) {
    print ("<tr bgcolor=#ddeeff>\n");
  } else {
    print ("<tr>\n");
  }
  print <<<EOF
    <td>$name</td>
    <td><input name="$val" type="text"
      value="$defval" size="40"></td>
EOF;
  print("</tr>\n");
      }
    }
                                                    ③
    if ($dbmodel->is_really_array(
      $dbmodel->multi_val_cols)) {
      foreach ($dbmodel->multi_val_cols as $col) {
  print <<<EOF
    <tr>
    <td>{$col_names[$col]}</td>
    <td>
EOF;
  foreach (
    $dbmodel->multi_val_values[$col] as $val) {
    print <<<EOF
      $val
EOF;
  }
  print("</td>\n");

  if (isset($dbmodel-
>multi_val_candidates[$col]) &&
      $dbmodel->is_really_array(
        $dbmodel->multi_val_candidates[$col])) {
    print<<<EOF
      <td>
      <select name="candidates[$col]">
EOF;
    foreach ($dbmodel-
>multi_val_candidates[$col] as $v) {
      print <<<EOF
        <option value="{$v["key"]}">{$v["val"]}
EOF;
    }
    print<<<EOF
      </select>
      </td>
      <td>
      <input type="submit"
        name="cand_reg[$col]" value="追加">
      <input type="submit"
        name="cand_del[$col]" value="削除">
      </td>
EOF;
  }
  print ("</tr>\n");
      }
    }
                                                    ④
    print <<<EOF
      </table>
      <input type="submit" name="first_page"
        value="検索開始">
      <input type="submit" name="new_first_page"
        value="新規追加検索開始">
      <input type="submit" name="next_page"
        value="次ページ">
      <input type="submit" name="previous_page"
        value="前ページ">
      <br>
      <input type="submit" name="update"
        value="更新">
      <input type="submit" name="insert"
        value="追加">
      <input type="submit" name="delete"
        value="削除">
      <input type="submit" name="return"
        value="メニューに戻る">
      </form>
EOF;
                                                    ⑤
    return $this->show_cols;
  }
}
?>
```

viewクラス（リスト4）は実際に画面を表示する処理を行います．col_names変数は，列名とそれを画面に表示する際の表示名の連想配列を格納しています．ここでまとめて表示名を管理することにより，複数の画面に使われてる列名を一括変更するなど，メインテナンス性が高まります．

search_form

唯一のメンバ関数で，すべての表示処理を司ります．引数は2つです．第一引数（$model）は，モデルクラスオブジェクトです．第二引数（$def）はフォームに表示するデフォルト値で，省略可能です．関数の戻値はこのクラスで定義されている表示列名の配列で，キーが列名，配列要素値が“TRUE”です．

まず，テーブルに関する列名などの「メタデータ」を取得します（リスト4-①）．そしてリスト4-②からフォームを作成してその中で列を表示します．

実際に列を表示するのはリスト4-③のforeachループです．ループの中で$arは連想配列で，$ar["name"]で列名が参照できます．

もしその列がshow_colsに含まれていたら，それを表示します．その際$defにデフォルト値が含まれていたら，それを表示します．また，その列がもしモデルクラスの$is_writable_col変数に含まれていたら，その列の更新が可能であることを明示するためにセルの色を変えます．

リスト4-④は著者またはキーワードの追加画面処理です．これについては次回で説明します．

そして最後に検索開始などの各種ボタンを表示します（リスト4-⑤）．

次回予告

次回は，残ったモデルクラス(dbaccess)と記事テーブル固有の処理について解説を行う予定です．