さわって実感

リレーショナルデータベース設計 6

アプリケーション設計

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

アプリケーションの設計

開発環境

前回までにテーブル設計とトランザクション設計が終わりましたので，今回はいよいよアプリケーションの設計に入ります．その前に，実際に開発に使う環境を確認しておきましょう[注1]．

●データベース管理システム

前回PostgreSQL 7.2がリリースされたことをお伝えしましたが，その後バグ修正バージョンであるPostgreSQL 7.2.1がリリースされているので，さっそく使ってみることにします．

● PHP

アプリケーション開発言語は，以前から予告している通りPHPを使います．本稿執筆時点（5月上旬）の最新版はPHP 4.2.0ですが，筆者が試した範囲では一部問題があったので，その1つ前のバージョンである4.1.2を使用します（これ以前のバージョンはセキュリティに問題があるので，そのまま使うことは推奨されていません）．

●Webサーバ

もちろん定番のApacheを使います．Apacheは現在2.0系列が開発されていますが，まだ十分安定しているとは言いがたいようなので，ここでは1.3系列の最新版1.3.24を使います．

アプリケーションフレームワーク

開発環境はすんなり決まりましたが，問題は「アプリケーションフレームワーク」です．アプリケーションフレームワークとは少々曖昧な言い方ですが，ここではプログラミング言語であるPHPとアプリケーションの間にはさまるライブラリやクラス，あるいは場合によってはアプリケーションサーバを指すものとします．

PHPの場合，各種フレームワークが利用できますが，一番広く使われているのはPHPLIBとPEARでしょう．どちらもPHPのクラスライブラリであり，DBアクセスの抽象化などのさまざまな機能を提供します．筆者の独断によれば，それぞれ以下のような特徴があると思います．

● PHPLIB

PHP3の頃から使われている実績のあるクラスライブラリです．PHP本体とは別に開発されています．DBアクセスの抽象化に加え，強力な認証機能を提供しているのが強みです．反面セッション管理を独自に実装しているため，PHP4組み込みの効率の良いセッショ

注1）筆者が確認のために使用した環境はVine Linux 2.1CRですが，他の環境でもさほど違いはないはずです．

ン管理を利用できません．

●PEAR

最近急速に充実してきたPHP4の標準クラスライブラリ群です．実にさまざまな機能がありますが，認証機能がないこと，あまりにも急速に進化しているためか満足なドキュメントがないのが欠点です．また，クラスによっては実装がまだまだ洗練されていないものもあります．

PEARも魅力的ですが，今回は筆者の使い慣れているPHPLIBを使用します．実はPHPLIBは，PHPのグローバル変数にフォームなどから入力された値が自動セットされる機能[注2]をオフにしている環境では動作しないという致命的な問題がありましたが，http://www.geocities.jp/rui_hirokawa/php/php-book2/にて廣川類氏が修正パッチを公開され，この問題が解決しました．本稿でもありがたく使わせていただくことにします．

なお，誌面の都合もあって，本稿ではApacheやPHPそのものについては解説しません．今やPHPの各種解説書は巷にあふれていますので，お好きなものを選んで参考にしてください．個人的には，最近発刊された通称『青マンモス本』[注3]を推薦しておきます．

システム構成

今回作成するシステムの構造を示します（図1）．

使用ツールのインストール

PostgreSQL

PostgreSQLのインストール方法は本連載の第1回で説明したので，ここでは省略します．そのときはPostgreSQL 7.1.2を使いましたが，7.2.1でもインストール方法は変わりません．7.1.2→7.2.1と読み替えていただければ結構です．ただし，7.1.xと7.2.xではデータベースの互換性がありませんので，7.1.xで使っていたデータベースは7.2.1ではそのままでは利用できません．7.1.xが動いている状態でpg_dumpallというツールを使ってデータを一旦バックアップし，7.2.1をインストール後，データを戻します．具体的には以下を実行します．

```
[7.1.xで実行]
$ pg_dumpall |gzip -c > /tmp/db.out.gz
[7.2.1をインストール後，以下を実行]
$ initdb
$ gunzip -c /tmp/db.out.gz|psql template1
```

いずれもPostgreSQLのスーパユーザ（通常postgresというユーザ）で実行します．なお，どうしても7.2.1のインストールが面倒だという方は，7.1.2をそのまま使っていただいても構いません．今回のアプリケーションは問題なく動作するはずです．

Apache

Apacheの最新ソースはhttp://www.apache.org/から辿れるミラーサイトで入手できます．アーカイブのファイル名はapache_1.3.24.tar.gzです．これを取得して，/tmpに置いておきます．

続いて，rootユーザで図2の手順を実行します．

次に適当なブラウザを使ってhttp://localhostというURLを表示して，図3のような画面が表示されれば

図1　作成するシステムの構造

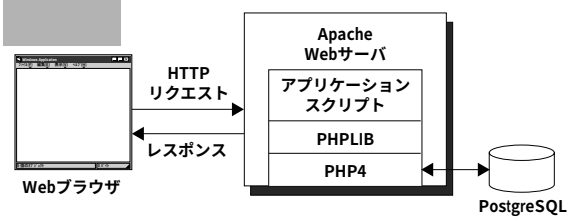


図2　Apacheのインストールと実行

```
# cd /usr/local/src
# tar xfz /tmp/apache_1.3.24.tar.gz
# cd apache_1.3.24
# env OPTIM="-O2" ./configure --enable-module=so
# make
# make install
# /usr/local/apache/bin/apachectl start
```

注2）一見便利なようですが，セキュリティ上の弱みになる可能性があり，現在は使用を推奨されていません．
注3）『PHP4徹底攻略 実践編』／廣川類，桑村潤，小山哲志／ソフトバンク／ISBN4-7973-1519-9．

Apacheのインストールは成功しています．

PHP

図4のようにコマンドを打ち込んでください．これで，PHP4モジュール/usr/local/apache/libexec/libphp4.soがインストールされ，/usr/local/apache/conf/httpd.confの205行目あたりに

```
LoadModule php4_module    libexec/libphp4.so
```

という行が書き込まれたはずです．確認してください．

次に，PHPの拡張子である“.php”をApacheに認識させるために/usr/local/apache/conf/mime.typesに以下の2行を追加します．

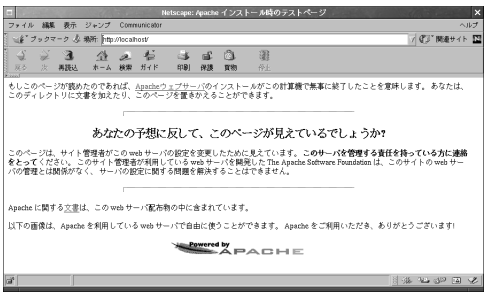
```
application/x-httpd-php php
application/x-httpd-php-source phps
```

さらに日本語を扱えるように，/usr/local/lib/php.iniを設定します．89行目に

```
output_buffering = Off
```

という行があるので，

図3　Apacheインストール後の初期画面




```
output_buffering = On
```

に変更します．その少し下に，

```
output_handler =
```

という行があるので，

```
output_handler = mb_output_handler
```

に書き換えます．また831行付近に

```
[mbstring]
;mbstring.internal_encoding = EUC-JP
;mbstring.http_input = auto
;mbstring.http_output = SJIS
;mbstring.detect_order = auto
;mbstring.substitute_character = none;
```

のような部分があるので，コメント（行頭の“;”）を外します．

最後にセキュリティを強化するために302行目の

```
register_globals = On
```

を

```
register_globals = Off
```

に変更します．

以上で最低限のセットアップができたので，さっそくPHPの動作確認を行います．PHPにはテストも兼ねてPHPの設定状態を表示する関数が付属していますので，それを呼び出すごく簡単なPHPスクリプトを作ります．

図4　PHPのインストール

```
# cd /usr/local/src
# tar xfz /mnt/cdrom/archives/php-4.1.2.tar.gz
# cd /usr/local/src/php-4.1.2
# ./configure  --without-mysql --with-apxs=/usr/local/apache/bin/apxs --enable-mbstr-enc-trans
   --enable-mbstring --enable-versioning --enable-trans-sid
  （実際には1行で入力します）
# make
# make install
# cp /usr/local/src/php-4.1.2/php.ini-dist /usr/local/lib/php.ini
```

```
<?php
  phpinfo();
?>
```

この3行だけのスクリプトを/usr/local/apache/htdocs/test.phpとしてセーブします．そして

```
# /usr/local/apache/bin/apachectl stop
# /usr/local/apache/bin/apachectl start
```

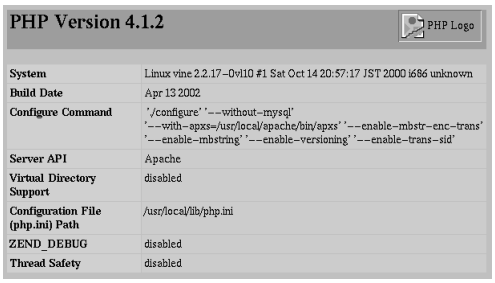
にてApacheを再起動します．適当なブラウザでhttp://localhost/test.phpというURLを開き，図5のようなページが表示されれば正常に動作しています．なお，右上の“PHP Logo”の部分の画像が壊れていますが，これは異状ではありません．むしろ，もしここにPHPのロゴが表示されていると日本語の処理がまともに動きません．そのときはphp.iniの設定を確認してください．

PostgreSQL用拡張モジュール

ここまでの設定では，PostgreSQLにアクセスするためのモジュールがPHPに組み込まれていません．PHP本体のコンパイル時に組み込んでしまってもよいのですが，それではPostgreSQLのバージョンアップ時に再びPHPをコンパイルしなければならなくなってしまい，柔軟性に欠けるので，あえて後付けでPostgreSQLモジュールをインストールするわけです．

rootで図6を実行することにより，PostgreSQL用の拡張モジュールpgsql.soが/usr/local/lib/php/extensions/pgsql.soにインストールされます．

図5　PHPの動作確認画面



しかし，これだけではまだPHPはこのモジュールを認識できません．PHPに新しいモジュールを認識させるためには，PHPの設定ファイルである/usr/local/php/lib/php.iniの修正が必要です．

以下のような行があります．

```
extension_dir = ./
```

これを以下のように書き換えます．

```
extension_dir = /usr/local/lib/php/extensions
```

そして，その後に次の行を追加します．

```
extension=pgsql.so
```

これで完了です．php.iniを書き換えたので，Apacheの再起動が必要です．

```
# /usr/local/apache/bin/apachectl stop
# /usr/local/apache/bin/apachectl start
```

PHPLIB

PHPLIBはPHPスクリプトで書かれているので，特にコンパイルなどは必要とせず，簡単にインストールできます．ただし，前述のように素のPHPLIBではなく，廣川氏のパッチが施されたものを使用します．

PHPLIBのインストールは，クラスライブラリの部分を単にコピーするだけで完了ですが，本稿では読者の便宜を図ってアプリケーションのスクリプトに同梱させることにしました．この方法では，PHPLIBを共有して使う場合に比べ，PHPLIBの細かなバージョンの違いによる動作の違いに悩まされずに済むメリットもあります．

次節で解説する手順でアプリケーションをインストールすると，phplib/というサブディレクトリがありますが，そこに入っているのがパッチ済のPHPLIBです．

図6　PostgreSQL用拡張モジュールのインストール

```
# cd ext/pgsql
# phpize
# aclocal
# ./configure
# make
# make EXTENSION_DIR=/usr/local/lib/php/extensions install
```

6 アプリケーション設計

アプリケーションのインストール

❶PHPスクリプトの展開

以下，ステップ❶と❷はrootで実行してください．

今回作成したアプリケーションは1000行以上あり，リストを見ながら打ち込むのは困難なので，WEB+DB PRESSのWebサイト（http://www.gihyo.co.jp/wdpress/）からダウンロードできるようにしてあります．まずそこからmagazine.tar.gzをダウンロードして/tmp/に置いてください．それを適当な場所に展開します．

たとえば，/usr/local/apache/htdocs/に展開したとすると，

```
# tar xfz /tmp/magazine.tar.gz -C /usr/local/apache/htdocs
```

となります．

❷Apacheコンフィギュレーションファイルの修正

次に，Apacheのコンフィギュレーションファイルを修正します．/usr/local/apache/conf/httpd.confの一番最後に，

```
<Directory /usr/local/apache/htdocs/magazine>
  AllowOverride Options
</Directory>
```

図7　nobodyユーザの登録

```
$ createuser nobody
Shall the new user be allowed to create databases? (y/n) n
Shall the new user be allowed to create more new users? (y/n) n
CREATE USER
```

図8　データベースの作成

```
$ createdb webdb
CREATE DATABASE
```

図9　データベースの初期化

```
$ cd /usr/local/apache/htdocs/magazine/support
$ psql -U nobody -f create.sql webdb
psql:create.sql:1: ERROR:  sequence "publishers_publisherid_seq" does not exist
psql:create.sql:2: ERROR:  table "publishers" does not exist
psql:create.sql:8: NOTICE:  CREATE TABLE will create implicit sequence 'publishers_publisherid_seq' for
SERIAL column 'publishers.publisherid'
psql:create.sql:8: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index 'publishers_pkey' for
table 'publishers'
[中略]
psql:create.sql:112: NOTICE:  ALTER TABLE will create implicit trigger(s) for FOREIGN KEY check(s)
ALTER
psql:create.sql:115: NOTICE:  ALTER TABLE will create implicit trigger(s) for FOREIGN KEY check(s)
ALTER
```

図10　テーブルの作成

```
$ psql -U nobody -f create_database.pgsql webdb
psql:create_database.pgsql:5: ERROR:  table "active_sessions" does not exist
psql:create_database.pgsql:12: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index
'active_sessions_pkey' for table 'active_sessions'
CREATE
psql:create_database.pgsql:14: ERROR:  table "auth_user" does not exist
psql:create_database.pgsql:20: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index
'auth_user_pkey' for table 'auth_user'
CREATE
psql:create_database.pgsql:22: ERROR:  table "auth_user_md5" does not exist
psql:create_database.pgsql:28: NOTICE:  CREATE TABLE / PRIMARY KEY will create implicit index
'auth_user_md5_pkey' for table 'auth_user_md5'
CREATE
CREATE
CREATE
INSERT 493356 1
INSERT 493357 1
```

を追加してください.

❸データベースの設定

ここから先はpostgresユーザで実行します.

PostgreSQL関連の設定を行います．まず，図7のようにnobodyユーザを登録します.

次に，今回アプリケーションが使うデータベースを作成します（図8）．ここでは“webdb”とします.

続いて，データベースを初期設定します（図9）．ERRORという文字が見えますが，これはまだ作成していないテーブルを念のために削除するときに出ているもので，気にする必要はありません.

最後に，PHPLIBが使用するテーブルを設定します（図10）.

◇ ◇ ◇

以上で，アプリケーションのインストールは終わりです.

作成するアプリケーション

今回作成するのは，前回までに作成した雑誌データベース（図11）を管理するためのアプリケーションです.

純粋のアプリケーションというよりは，管理ツールに近い位置付けで，出版社（publishers），雑誌(magazines)，発刊雑誌（iss_mags），記事（articles），著者（writers），キーワード（keywords）の6つのテーブルに対して，検索，追加，変更，削除操作ができます.

記事と著者，記事とキーワードを結び付ける2つのテーブル（writingsとarticles_keywords）に関しては，直接更新操作を行いません．これらのテーブルは中身が数値のIDであり，人間が直接編集を行うのは困難だからです．その代わり，記事テーブルの編集画面で登録済の著者やキーワードから必要なものを選択できるようにします.

◇ ◇ ◇

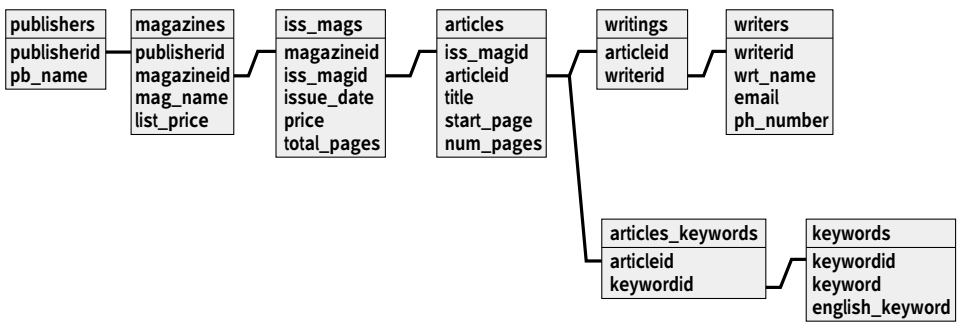
それでは，アプリケーションの全体の流れを掴むために，順を追って画面を見ながら概要を説明します.

ログイン画面（図12）

登録ユーザだけがこのアプリケーションを使えるようにするために，パスワード認証をかけます．PHPLIBでは，基本的な認証機能のクラスが提供されており，ユーザ認証の機能が簡単に実装できます．また，HTTPの基本認証に頼らず，独自の認証を行っていますので，ログイン／ログアウト制御も実装できますし，認

図12　ログイン画面

図11　前回までに作成した雑誌データベース



証画面のデザインも自由です．ユーザ管理のためのテーブルは，PHPLIB独自のものを使います．今回は組み込んでいませんが，パスワードの文字列が生で流れないようにすることもできますし，ユーザの追加削除を行う機能を追加することも可能です．

このサンプルアプリケーションでは，ユーザ認証の初期データとしてPHPLIBのサンプルに付属するデータをそのまま使っています．ユーザ名“kris”，パスワード“test”としてログインしてください．

本来ならばユーザ管理機能も作成すべきなのですが，今回はそこまでしていません．PHPLIBがユーザ管理に使用しているテーブルはauth_userです．手動でユーザを追加する場合は，create_database.pgsqlの最後にあるSQL文，

```
INSERT INTO auth_user VALUES ('c14cbf141ab1b7c
d009356f555b607dc','kris','test','admin');
```

を参考にしてください．データの意味は左から順に，

- ユーザID
  ユニークなユーザ識別子．PHPで書けば，md5(uniqid(HASH_SECRET))で生成できます．なお，HASH_SECRETは適当な文字列です．
- ユーザ名
  ログインのときに使用するユーザ名です．
- パスワード
  ログインのときに使用するパスワードです．
- ユーザ権限
  今回は使用していません．一律‘admin’でOKです．

となります．

なお，PHPLIBでは，デフォルトで認証の有効期間を15分にしています．したがってアプリケーションの画面を開いてから15分以上何もしないでいて，次に何か操作を行おうとすると，このログイン画面が表示されて再び認証を求められます[注4]．

初期メニュー画面（図13）

ここでは，編集の対象となる出版社（publishers），雑誌（magazines），発刊雑誌（iss_mags），記事（articles），著者（writers），キーワード（keywords）の6つのテーブルを選択します．

サンプルアプリケーションの使用を中止する場合は，「ログアウト」を選択します．

◇ ◇ ◇

それでは，各テーブル，画面について，説明を続けます．

出版社編集画面（図14）

この画面の下部にはたくさんボタンが並んでいます．これらのボタンは，他のテーブルの編集画面でもまったく同じ機能を持っています．以下，各ボタンごとに機能を説明します．また参考のために，アプリケーション内部で実行されるSQL文の例の一部を示します．

ここでは，データベースに対する操作のイメージを掴んでください．他のテーブルでも同様のSQL文（実際にはもっと複雑ですが）が使用されます．

図13 初期メニュー画面

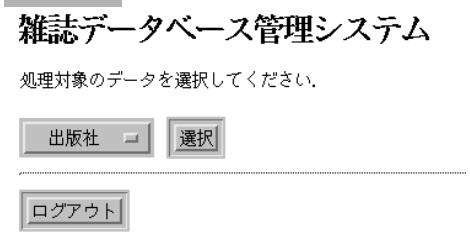
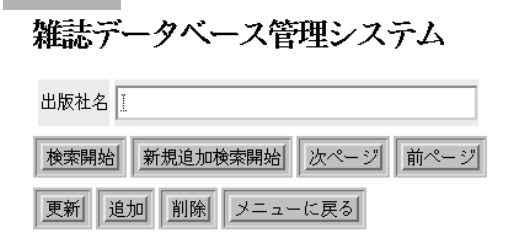


図14 出版社編集画面



注4）ちょっとわずらわしいような気もしますが，席を立っている間に他の人に勝手に画面を操作されないための考慮です．スクリーンセーバーのようなものだと思えば良いでしょう．ちなみに設定によって，このタイムアウト時間の15分をもっと長くしたり，あるいはまったくタイムアウトをかけないようにすることもできます．

●「検索開始」ボタン

出版社名からpublishersテーブル検索してその結果を表示します．検索キーは上のほうにある「出版社名」と書かれたボックスに入力します．今のところ完全一致検索しか許していません．

検索時に実行するSQL文の例を示します．

```
SELECT pb_name,publisherid,ctid FROM publishers
WHERE pb_name = '技術評論社' LIMIT 1 OFFSET 0
```

ここで，“LIMIT 1 OFFSET 0 ”の部分はPostgreSQL固有の拡張構文で，検索結果の中から先頭（OFFSET 0）の1行のみ（LIMIT 1）を表示します．

●「新規追加検索開始」ボタン

この機能はpublishersテーブルでは意味を持たないので，別途説明します．

●「次ページ」ボタン

「検索開始」によって複数行の結果が得られた場合，次の行を表示します．この機能はSELECT文の後に“ LIMIT 1 OFFSET n ”（nはページ番号）を付加することによって実現されています．

●「前ページ」ボタン

「次ページ」によって進んだページを前に戻します．ここでもOFFSETを使っています．

●「更新」ボタン

現在のデータを修正します．更新時に実行するSQL文の例を示します．

```
BEGIN;
LOCK TABLE publishers IN SHARE ROW EXCLUSIVE
MODE;
SELECT ctid FROM publishers WHERE publisherid
= '6';
UPDATE publishers SET pb_name = '(株)PostgreSQL
出版' WHERE ctid = '(0,12)'
COMMIT;
```

前号で説明したように，更新しようとした行が他のユーザに削除されてしまわないように，テーブルをロックしています．

●「追加」ボタン

新しく行を追加します．追加時に実行するSQL文の例を示します．

```
BEGIN;
LOCK TABLE publishers IN SHARE ROW EXCLUSIVE
MODE;
SELECT ctid FROM publishers WHERE pb_name =
'PostgreSQL出版';
INSERT INTO publishers (pb_name) VALUES(
'PostgreSQL出版');
COMMIT;
```

●「削除」ボタン

現在表示されている行を削除します．削除時に実行するSQL文の例を示します．

```
BEGIN;
LOCK TABLE publishers IN SHARE ROW EXCLUSIVE
MODE;
SELECT ctid FROM publishers WHERE publisherid
= '6';
DELETE FROM publishers WHERE ctid = '(0,15)';
COMMIT;
```

●「メニューに戻る」ボタン

再び編集対象テーブルを選択する初期メニューに戻ります．

では，さっそく出版社データを登録してみましょう．「出版社名」のボックスに「技術評論社」と入力し，「追加」ボタンを押します．見た目は何も変わりませんが，これで「技術評論社」が登録されています．このとき，publishersテーブルの主キーであるpublisheridはPostgreSQLのSERIAL型の機能により，自動的に採番されるので，入力の必要はありません．

出版社が1つだけではさみしいので，「PostgreSQL出版」という架空の出版社も登録しましょう．それから「出版社名」のボックスを空白にし，検索開始を押

します．検索条件が空白の場合は全件検索となり，「技術評論社」と「PostgreSQL出版」が検索されることになるので，まず「技術評論社」が表示されます．「次ページ」を押せば，今度は「PostgreSQL出版」が表示されるはずです．

ここで「PostgreSQL出版」を「(株)PostgreSQL出版」に書き換え，「更新」ボタンを押します．見た目は変わりませんが，出版社名が更新されています．「削除」ボタンを押します．すると「(株)PostgreSQL出版」が削除されます．「出版社名」のボックスが空白になったので，「検索開始」を押して全件検索を実行し，確かに「(株)PostgreSQL出版」が削除されていることを確認してください．

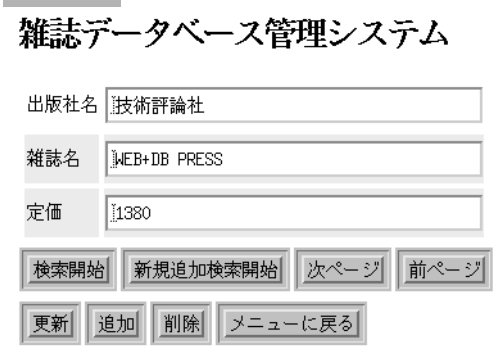
確認ができたら「メニューに戻る」を押して初期メニューに戻り，今度は雑誌テーブルの編集画面を表示します．

雑誌編集画面（図15）

この画面では，雑誌（magazines）テーブルの編集を行います．「出版社名」「雑誌名」「定価」の3つのボックスが表示されています．この中で「出版社名」だけは網掛け（実際には薄いブルーがかかっています）になっていませんが，これには理由があります．

magazinesテーブルには，「出版社名」ではなくて「出版社ID」が登録されていますが，IDを表示しても人間にはわかりにくいので，publishersテーブルから「出版社名」を引っ張ってきています．つまり，「出版社名」だけは網掛けにしないことによって，magazinesテーブルに所属しないデータであることを明示しているわけです．本アプリケーションでは，網掛けになっていないボックスのデータは検索キーには使えますが，更新の対象となりません．このことは他の画面でも同じですので，覚えておいてください．

図15　雑誌編集画面



さて，今のところmagazinesテーブルには何もデータが登録されていないので，「検索開始」を押しても何も出てきません．そこで新たに雑誌データを登録するのですが，そのためには，雑誌が「所属」している出版社を決める必要があります．そこで役に立つのが「新規追加検索開始」です．このボタンを押すと，「上位」のテーブルを検索し，網掛けになっていないボックスに表示します（もちろんこのボックスに検索条件を入れておけばそれも考慮されます）．

ここではさきほど登録した「技術評論社」が表示されるはずです．早速雑誌データを登録しましょう．ここでは，WEB+DB PRESSのデータを登録します（図15）．

発刊雑誌編集画面（図16）

この画面では，発刊雑誌（iss_mags）テーブルの編集を行います．雑誌編集画面と同様に網掛けのボックスはiss_magsに実体がある項目ですが，網掛け

図16　発刊雑誌編集画面

でないボックスは「上位」テーブルであるmagazinesとpublishersから持ってきています．

「新規追加開始」ボタンを押すと，先ほど登録したWEB+DB PRESSのデータが表示されるはずです．ここでは，本連載が開始されたVol.4（2001年9月15日発刊）のデータを登録します（図16）．

記事編集画面（図17）

ここでは，記事（articles）テーブルの編集を行います．発刊雑誌編集画面と同様に網掛けのボックスはarticlesに実体がある項目ですが，網掛けでないボックスは「上位」テーブルであるiss_mags，magazinesとpublishersから持ってきています．

「新規追加開始」ボタンを押すと，先ほど登録したWEB+DB PRESS Vol.4のデータが表示されるので，本連載開始時の記事のデータを登録します（図17）．

さて，この画面には「著者名」と「キーワード」という項目があります．それぞれこの記事を書いた人（達）と記事に関連したキーワードです．図11をもう一度見てください．これらは，それぞれwritingsとarticles_keywordsを経由して，著者（writers），キーワード（keywords）と関連づけられていますが，まだこれらのテーブルが空なので何もデータが表示されていないのです．

まず著者（writers），キーワード（keywords）にデータを登録してから再びこの画面に戻ってくることにしましょう．

著者編集画面（図18）

この画面では，著者（writers）テーブルの編集を行います．特に上位のテーブルはないので，全部網掛けの項目になっています．ここでは，勝手ながら私のデータを登録します．

キーワード編集画面（図19）

この画面では，キーワード（keywords）テーブル

図17　記事編集画面

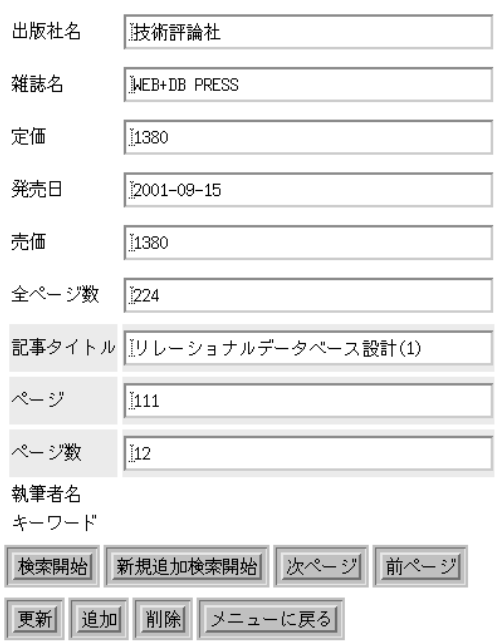


図18　著者編集画面

図19　キーワード編集画面

の編集を行います．特に上位のテーブルはないので，全部網掛けの項目になっています．ここでは，「PostgreSQL」と「関係データベース」をキーワードとして登録します．

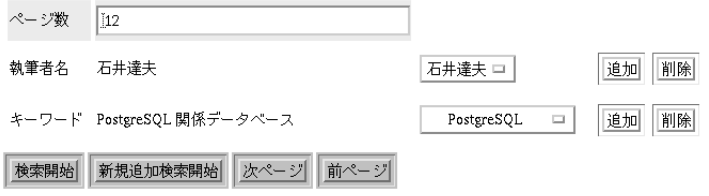
著者とキーワードの登録

記事テーブルのところで説明したように，著者，キーワードの登録に戻りましょう．メニュー画面から「記事」を選び，続いて「検索開始」を選んで記事テーブルの全件検索を行います．すると，さきほどと異なり，「執筆者名」「キーワード」の右にプルダウンメニューと「追加」「削除」のボタンが表示されます（図20）．

ここでメニューから著者あるいはキーワードを選択して追加を選んでください．すると，この記事に関連づけられた著者名，キーワードが表示されるようになります（図21）．

図20　著者とキーワードの登録

図21　記事，著者，キーワードの関連づけ



アプリケーションの設計方針

一通りアプリケーションのイメージを掴んでいただいたところで，設計，実装について説明します．

MVCモデルの採用

この手のアプリケーションでは定石となっているMVCモデルを採用します．MVCモデルとは，Model，View，Controllerの3つにプログラムを分割する手法です（図22）．

Modelは業務ロジックを扱います．今回はDB管理ツールですので，DBを操作するロジックから構成されることになります．

Viewは画面表示を行います．Webアプリケーションでは，HTMLを表示する部分です．Viewはその中に業務ロジックや表示データの生成を含みません．表示に必要な情報はすべてModelから入手します．

最後にControllerですが，ここはModelやViewを制御する部分です．Webアプリケーションでは，ユーザからのリアクション（検索開始など）を受けて，適切なModelやViewを起動する役割を果たします．

MVCモデルを採用することにより，プログラムの見通しが良くなる他，業務ロジックの再利用が可能になる上，Viewに業務ロジックを含む必要がなくなるので，ロジックと画面デザインの分離が可

能になるなどのメリットがあります．

MVCモデルはJavaによるWebアプリケーションの世界ではもはや常識となっていますが，なぜかPHPの世界ではあまり使われていないようです．しかし，PHPでも十分MVCの恩恵に預かることは可能ですので，もっと広く使われても良い手法だと思います．

クラスの積極的な採用

今回管理の対象となるテーブルは全部で8つもあり，個々のテーブルごとにコードを書いていてはきりがありません．そこで，プログラムロジックはできるだけテーブル構造に依存しないようにして基底クラスを作成し，個々のテーブルに関するクラスはこの基底クラスを継承します（図23）．

DBに関するロジックのうちpublishersとmagazinesに見られるような「親子関係」（1：Mの関係），またarticles - writings - writersのような1：M：1関係に関する検索／更新ロジックは，基本的なロジックとして基底クラスに取り込んでいます．

従来単一のテーブルを扱う汎用的なライブラリはありましたが，このように複数のテーブルにまたがる処理を一般的に扱うライブラリはあまり例がなかったと思います．まだまだ機能も足りず，洗練されているとは言えない実装ですが，この基底クラスは本連載の雑誌データベースに依存したところはありませんので，他の目的にも利用できるはずです．読者の皆さんも，用途に合わせて改造して利用すると良いでしょう[注5]．

次回予告

次回も引き続きサンプルアプリケーションの解説を行います．ソースコードおよび使用するSQL文の詳細な解説になる予定です．

図22　MVCモデル

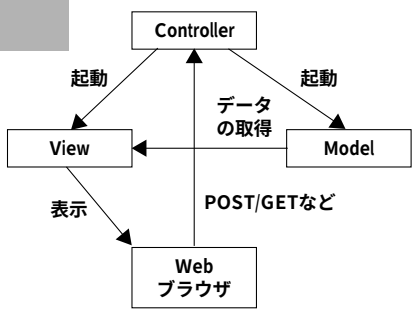
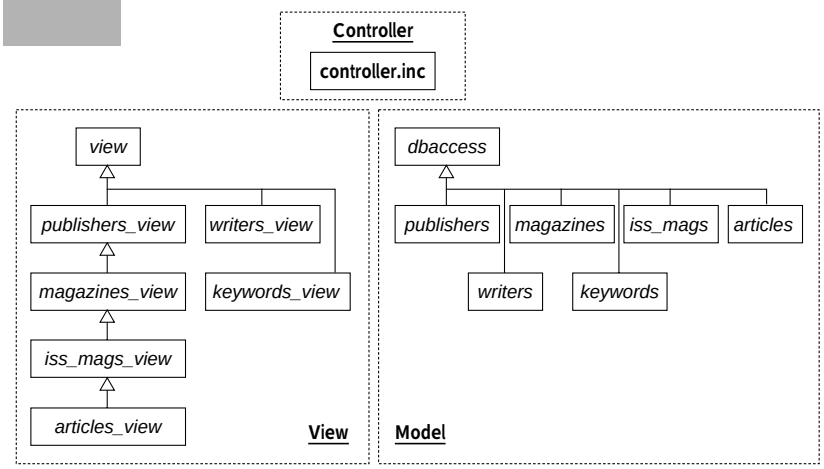


図23　アプリケーションのクラス図



注5）使用許諾条件については，ソースに同梱されるLICENSEファイルをお読みください．基本的にBSDライセンスです．