PDO_4D ドライバー

By Jesse Pina, Technical Services Team Member, 4D Inc.

Technical Note 10-12

概要

PDO_4D ドライバーは 新しいプログラムで、Apache や IIS などの Web サーバーから 4D データベースへの PHP を使用した接続を容易にします。4D は PDO_4D ドライバーの開発のために、オープンソースプロジェクトのスポンサーとなりました。PDO_4D ドライバーは PHP を使用する Web サーバーにインストールすることができ、SQL コードや 4D メソッドを実行させるために使用できます。このテクニカルノートの目的はシステムを構成する部品、それらがどのように動作するのか、そして PDO_4D ドライバーの使用方法を説明することです。例題および PHP に関する基本的な情報も提供します。

はじめに

4D は PDO ドライバーを作成するためにオープンソースプロジェクトのスポンサーとなりました。作成されたドライバーは PDO_4D と呼ばれ、これを使用することで Apache や IIS などのサーボパーティ Web サーバーから PHP を使用して 4D データベースにアクセスできます。PDO ドライバーの作成は、4D に SQL エンジンが実装されたことで可能となりました。PDO_4D ドライバーにより、PHP コードで SQL を使用して、4D データベースの SQL サーバーにアクセスできます。

サードパーティ Web サーバーから 4D データベースへのアクセスがより容易になったという事実に加え、PDO_4D ドライバーはコーディングの分業も可能にします。PDO_4D により、あなたは 4D のエクスパートとしてデータベースのコーディングにフォーカスし、フロントエンド PHP のコーディングは PHP のエクスパートに任せることができます。

このテクニカルノートでは、関連するいくつかのテクノロジーについてその概要とそれらがどのように動作するのか、そしてどのように PDO_4D を使用するのかを説明します。例題および PHP に関する基本的な情報も提供します。またサードパーティの Web サーバーを使用したことが無いという方のために、Apache と PHP、そして PDO_4D をセットアップするステップを説明します。

PDO とは

説明

まず http://PHP.net にある説明を示します:

- PHP – 広く使用されているオープンソースの汎用スクリプトランゲージで、特に Web の開発に適しており、HTML に組み込むことができます。
- PDO – PHP Data Objects (PDO) 拡張は、PHP によるデータベースへのアクセスに軽量で一貫性のあるインターフェースを提供します。PDO インターフェースを実装するそれぞれのデータベースドライバーはデータベース特有の機能を拡張機能として公開できます。PDO 拡張を使用してデータベースの機能を実行する

ことができるというわけではない点に留意してください。データベース専用の PDO ドライバーを使用してデータベースサーバーにアクセスします。

PDO は PHP の拡張であり、データアクセスの抽象レイヤーを提供します。言い換えると、データベースシステム固有のコードを取り除き、基本的に SQL 文を書くように PHP コマンドを簡略化します。どのようなデータベースシステム (4D, MySQL, Oracle, SQLite, ...) が使用されていても、インターフェースは同様です。ここでインターフェースという用語はデータベースにアクセスするために書かれた実際の PHP コードを意味します。PDO 拡張は基本的に、インターフェース (PHP コード) を変えずに、データベース固有のドライバーを構築することを可能にします。

特定の (ここでは 4D) データベースシステムにアクセスするには、専用の PDO ドライバーが作成される必要があり、これが PDO_4D オープンソースプロジェクトの目的です。

利用

PDO の主なゴールの一つは、データベース特有のコードを書かずに、さまざまなデータベース(4D, MySQL, SQLite, ...)にアクセスするコードを開発者が書くことを手助けすることです。実際の接続情報や利用可能なストアドプロシージャーを例外として、アクセスされるデータベースに関わらず同じ PHP コードが動くようにします。また PHP コードは変更されなくとも、データベースごとの SQL 実装は異なることがあります。なので SQL コードが異なる場合もあります。

Web アプリケーション開発時に共通して発生する問題に、可能な限りデータベースレイヤーを抽象化するために多大な時間と努力が必要であるということがあります。それに加えて、多くのデータベース固有のコードを書かなくてはなりません。PDO は基本的にデータベースアクセスの抽象化レイヤーを提供するので、開発者はそれを行う必要がありません。

PDO の利用者

先に進める前に述べておきたいと思いますが、PDO_4D ドライバーは 4D の開発者様よりもどちらかというと PHP の開発者様に使っていただくことを意図しています。PDO_4D ドライバーを使用してコードを書くために必要なことは基本的に: 1) PHP に関する基本的な知識; 2) SQL の基本的な知識、です。Web ページを構築するために PHP を知っている必要があり、データベース要素にアクセスするために SQL を知っている必要があります。

PDO_4D の利点の一つは、データベースコーディングから Web コーディングを分離できることにあります。4D デベロッパーはバックエンドのコーディングにフォーカスし、PHP デベロッパーはフロントエンドのコーディングにフォーカスできるようになるのです。

プロジェクトの詳細

PDO_4D オープンソースプロジェクトは現在ソースコードのみを利用可能としています。実際のバイナリーは PHP グループまたは PECL ツールから提供される予定です。このテクニカルノートの例題で使用されている PHP ドライバーは 2009 年 9 月 1 日にリリースされたバージョン 0.2.1 のソースを使用してビルドされています。 このソースあるいは他のバージョンのソースは以下からダウンロードできます:
http://pecl.php.net/package/PDO_4D

ソースコードのほかに、パッケージにはどのように PDO_4D ドライバーをビルドするかの方法とライセンス情報も含まれています。

PDO_4D: 動作環境、設定、そしてインストール

PDO_4D は PHP から 4D データベースにアクセスできるようにするために開発されました。先に示した通り、PDO_4D は PDO インターフェースを実装し、4D データベースにアクセスする PHP コードを開発者が書くことを手助けします。

動作環境

PDO_4D ドライバーを使用するには、以下の 3 要素が必要です:

- SQL サーバーが起動されている 4D v12 アプリケーション
- Apache や IIS など PHP をサポートする Web サーバー
- PHP バージョン 5.2.0 以降および mbstring モジュールと PDO 1.0.0 以降

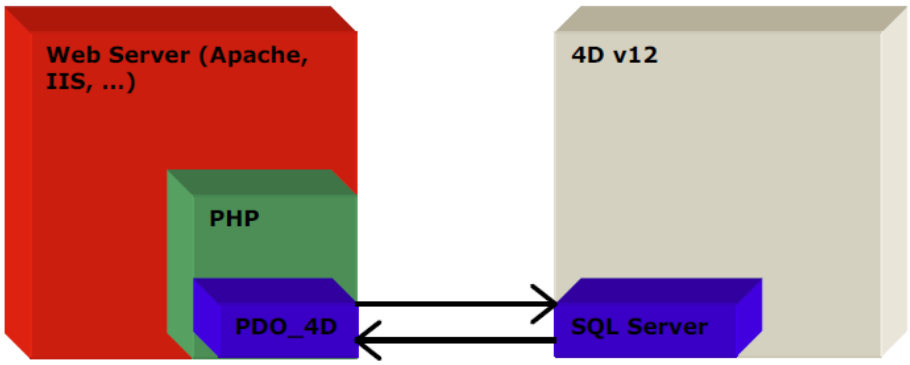
Note　　4D データベースは Apache や IIS と異なるマシンで実行することができます。

上記動作環境に適合する組み合わせは多数あります。例えば PHP をサポートする Web サーバーには Apache と Internet Information Server (IIS) があります。また Apache をサポートする OS には Mac OS、Windows、そして Linux などがあります。いくつかのシステムには Web サーバーと PHP がデフォルトでインストールされていますが、手作業で部品をインストールしなければならない場合もあります。Apache や PHP を実行できるサードパーティーの MAMP、XAMPP、WAMPServer などの製品もあります。どのような構成を選択するかはあなたの選択とニーズに基づきます。ここでのポイントはいろいろと選択肢があるということです。

設定 (どのように動作するのか)

PHP コードを含む Web ページは Web サーバーで管理されます。PHP コードが解釈されるためには、Web サーバーに PHP モジュールがインストールされていなければなりません。PHP モジュールは Web サーバーに既にインストールされている場合もあれば、手作業でインストールする必要がある場合もあります。

PHP モジュールをインストールしたら、PDO_4D 拡張を追加できます。この時点で、PDO_4D ドライバーを呼び出す PDO コマンドを含む、4D データベース内の SQL サーバーと通信する PHP コードを書くことができます。



インストール

ここではサポートされるセットアップのうち一つのインストール処理について詳細を説明します。このセットアップは推奨されるという意味ではありません。単にこのテクニカルノートで例題として使用されるということで、この例を基に PDO_4D ドライバーの利用方法を説明します。

環境

例で使用するセットアップの詳細やバージョンは以下の通りです:

- マシン: MacBook Pro
- CPU: Intel Core 2 Duo
- OS: Mac OS X, 10.5.8
- Web サーバー: Apache 2.2.11
- PHP: 5.2.8
- PDO_4D, バージョン 0.2.1 のソースを使用したビルド

ステップ

Apache Web サーバーで PHP を有効にします:

- テキストエディタを使用してファイル"/private/etc/apache2/httpd.conf"を開き、以下の行のコメントを外します:
  ```
  # LoadModule php5_module libexec/apache2/libphp5.so
  ```
  つまりこの行は以下のようになります
  ```
  LoadModule php5_module libexec/apache2/libphp5.so
  ```
- "/private/etc/php.ini.default"を"/private/etc/php.ini"にコピーします。

次に PDO_4D ドライバーをインストールします:

- テキストエディタを使用してファイル"/private/etc/php.ini"を開き、以下のコード行を追加します:

```
extension=pdo_4D.so
```

- "PDO_4D.so"ファイルを PHP の"extensions"フォルダーに配置します。

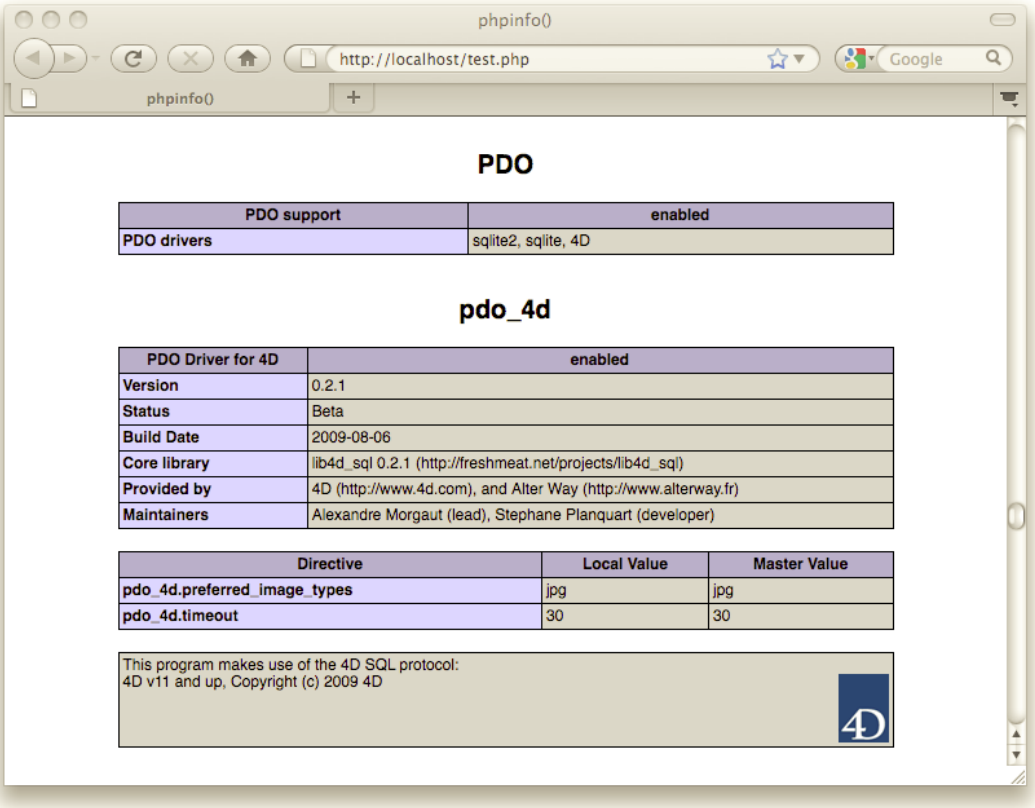
Note　このフォルダーの場所は php.ini ファイルで指定されているので、このファイル内で"extension_dir"から始まる行を探してください。このセットアップでは、"extensions"という名前の新しいフォルダーを作成し、すべての PHP 拡張をまとめて配置するようにしました。この新しいフォルダーを使用するために、この行を以下のように変更しました: extension_dir extension_dir = "private/etc/extensions"

最後に、Apache Web サーバーを開始します: (これを行うには 2 つの方法があります)

- Mac OS システム環境設定で共有を選択し、Web 共有を有効にします。
- ターミナルを開き、以下のコマンドを実行します:

```
sudo apachectl start
```

この時点で PDO_4D がインストールされた PHP が有効な状態で、Web サーバーが開始されます。PDO_4D ドライバーを使用した PHP Web ページを提供できます。PDO_4D ドライバーが正しくインストールされているかは、phpinfo()関数で検証できます (後述の Tip 参照)。以下の情報が表示されます:



Tips/注意点

例題に進む前に、Apache Web サーバーを始めて使用する方のために、インストールをスムースに進めるための Tip と注意点をあげます。

- "phpinfo()"関数のみを実行するシンプルな PHP Web ページを作成します。ページの内容は以下のようになります:

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

このページをブラウザーで開くと、PHP の状態に関するカレントの情報を表示します。情報にはオプション、拡張、PHP バージョン、Web サーバー、環境、さまざまなファイルへのパス等が含まれます。

- 以下の 3 つの UNIX コマンドを使用して素早く Apache サーバを開始、停止、再起動ができます。

```
sudo apachectl start
sudo apachectl stop
sudo apachectl restart
```

- 管理者でログインしていても、上記の設定ファイルにアクセスするには多くの場合認証が必要です。このため"SUDO"が使用されています。

- php.ini ファイル内でエラーのログを有効にしているか確認してください。以下の行を探します:

```
log_errors = on
```

これによりすべてのエラーは"/private/var/log/apache2/error_log"ファイル (デフォルトの位置) に記録されるようになります。予期しないエラーが発生した場合、まずこのファイルを参照します

- Web ページを作成する際、ファイルに正しいアクセス権が設定されているかを確認します。Web ページを開く際にブラウザーにアクセス権エラーが表示される場合は、Web ルートフォルダー内のファイルのアクセス権をチェックします。このフォルダー内のすべてのファイルのアクセス権を設定する簡単な方法はターミナルから Web ルートフォルダーをカレントディレクトリにして以下のコマンドを実行します:

```
chmod 755 *.*
```

- PHP は大文字小文字を区別します。$myvar と$myVar は異なる変数となります。

PDO_4D を使用する

Web サーバーとデータベースの設定ができたら、データベースにアクセスする PHP コードを書くことができます。これは SQL コードと 4D メソッドが実行できるようになることを意味します。

SQL コードの実行

PDO_4D ドライバーのインストールが終了したので、次は SQL コードを使用して 4D データベースにアクセスするシンプルな Web ページを作成します:

- 新しい 4D データベースを作成し、SQL サーバーを開始します。
- "/library/webserver/documents"に test.php という名前の PHP ファイルを作成します。これはデフォルトの Web ルートディレクトリで、http.conf ファイルで変更できます。変更したい場合、以下の行で行います:

```
DocumentRoot "/Library/WebServer/Documents"
```

- test.php ファイルに以下の PHP コードを記述します:

```
<?php
$dsn = '4D:host=localhost;port=19812;charset=UTF-8';
$user = 'Administrator';
$pswd = 'test';
$db = new PDO($dsn, $user, $pswd);
$db->exec('CREATE TABLE IF NOT EXISTS myTable(id INT NOT NULL, value VARCHAR(100))');
unset($db);
echo 'done'; // この文字が表示されたら、コードは正しく実行された
?>
```

- ブラウザーで http://localhost/test.php にアクセスします。結果のページには"done"と表示されます。4D データベースを確認すると、"myTable"テーブルが作成されているはずです。

この例で行われたことは以下の通りです:

```
$db = new PDO($dsn, $user, $pswd);
```

この行は PDO_4D ドライバーを呼び出し、データベースへの新しい接続を作成します。$dsn、$user、$pswd 変数の情報は、適切なデータベースに接続するために PHP が使用します。ここでは 4D をデータベースとして使用するため、これらの変数には 4D データベースが実行されているマシンの IP アドレス、SQL サーバーの公開ポート番号、4D データベースにアクセスするためのユーザー名とパスワードを格納します。

```
$db->exec('CREATE TABLE IF NOT EXISTS myTable(id INT NOT NULL, value VARCHAR(100))');
```

この行は PDO_4D ドライバーを呼び出し、"myTable"テーブルを作成する SQL コードを実行します。

Note　　呼び出す SQL コードは 4D の SQL エンジンで実行可能なコードでなければなりません。

```
unset($db);
```

Unset は PHP のコマンドで、変数を解放します。

```
echo 'done'; // if you see this then the code ran successfully
```

echo は PHP のコマンドで、文字列を出力します。

4D メソッドの実行

特別な SQL シンタックスを使用して 4D メソッドを実行できます。これを行うには以下のようにします:

- 先と同じデータベースに"myAdd"メソッドを作成します。
- メソッドの内容は以下の通りです:

```
C_LONGINT($1;$2; $0)
$0:= $1+$2
```

- このメソッドのプロパティで、"SQL で利用可"を選択します。
- test2.php という名前の PHP ファイルを"/library/webserver/documents"フォルダーに作成します。
- test2.php ファイルの PHP コードは以下の通りです:

```
<?php
$dsn = '4D:host=localhost;port=19812;charset=UTF-8';
$user = 'Administrator';
$pswd = 'test';
$db = new PDO($dsn, $user, $pswd);
$stmt = $db->prepare('SELECT {FN myAdd(1, 3) AS INT } FROM _USER_SCHEMAS LIMIT 1');
$stmt->execute();
$results_array = $stmt->fetchAll();
echo 'The result of the addition is: ' . $results_array[0][0] .'<br>';
unset($stmt);
unset($db);
?>
```

- ブラウザーで http://localhost/test2.php にアクセスします。結果のページに"The result of the addition is: 4"が表示されます。4 は 4D メソッドから返された値です。これは単純な例ですが、4D メソッドに引数を渡して呼び出し、値を受け取る方法を示しています。

まず接続するためのコードは最初の例題と同じであることが分かります。そして 4D メソッドが特別な SQL シンタックスを使用して実行されます。しかしそれを行うには、SQL コードが事前に準備される必要があります。

```
$stmt = $db->prepare('SELECT {FN myAdd(1, 3) AS VARCHAR } FROM _USER_SCHEMAS LIMIT 1');
$stmt->execute();
```

$db->prepare コマンドでステートメントを準備し、$stmt->execute() コマンドで SQL コードを実行します。先の例題では myAdd メソッドに 2 つの引数を指定し、戻り値を VARCHAR と定義しました。

```
$results_array = $stmt->fetchAll();
echo 'The result of the addition is: ' . $results_array[0][0] . '<br>';
```

このコードで、一行目は実行された SQL コードから戻された値を受け取り、二次元配列に格納しています。そして次の行で二次元配列から結果を取り出し、メソッドの結果を出力しています。fetchAll() から返されるフォーマットと、個々の配列項目へのアクセスについては CRUD の例題でさらに説明します。

例題 – CRUD

CRUD はデータベースの基本機能を表すためにしばしば使用される略語で、それぞれ Create、Read、Update、Delete を表します。特にレコードのコンテキストではレコードの作成や読み出しなどを指します。このテクニカルノートに含まれる例題では 4 つの PHP ページがあり、それぞれ CRUD 処理を実装しています。その他の処理としてはテーブルの作成や削除が組み込まれています。

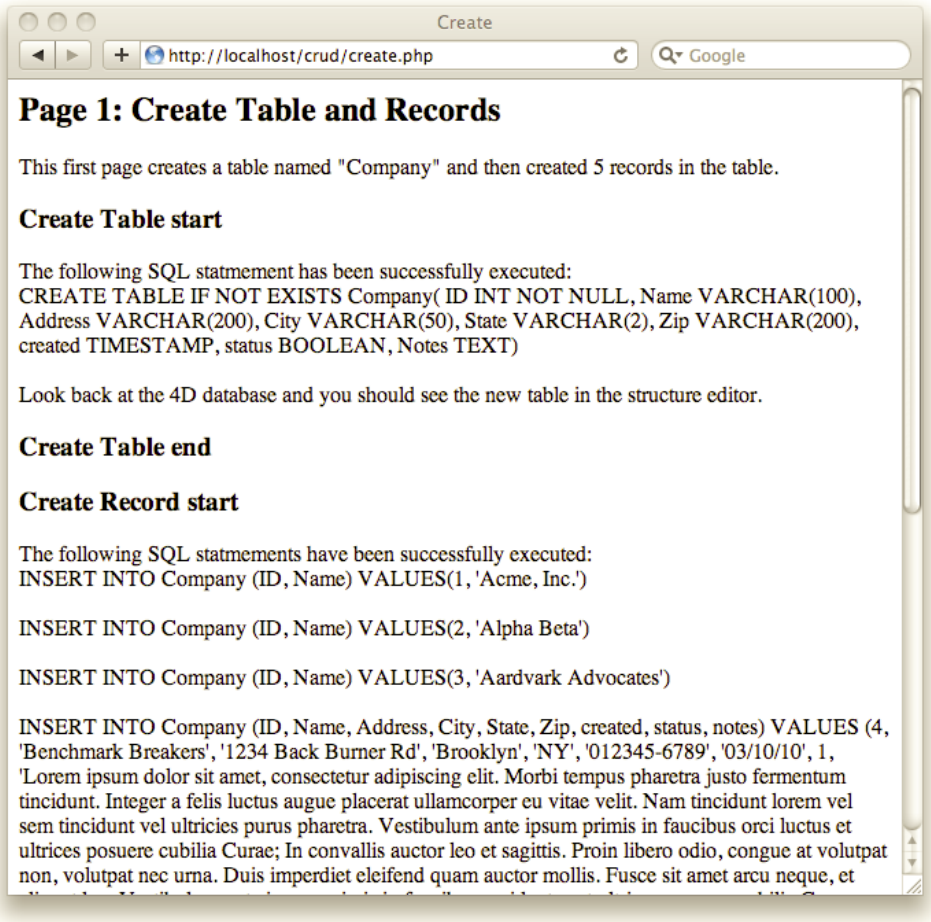
例題を使用するには以下のステップに従ってください:

- "CRUD.4dbase"データベースを 4D v12 で開きます (SQL サーバーが自動で起動されます)。
- Web サーバーを設定し、PDO_4D ドライバーをインストールします (4D データベースと同じマシンにするとやりやすいかもしれません)。
- Web サーバーを開始します。
- 添付の"CRUD"フォルダーを Web サーバーのルートフォルダーに配置します。先の例題では "/library/webserver/documents"になります。

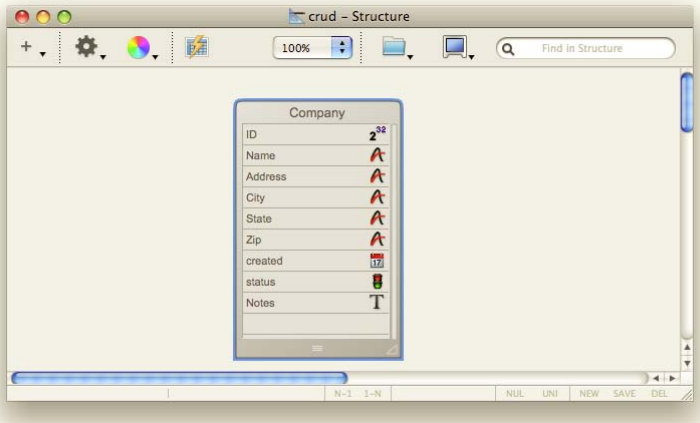
Note　　このテクニカルノートに含まれる例題データベースは、最初に示した動作環境の下で作成されました。

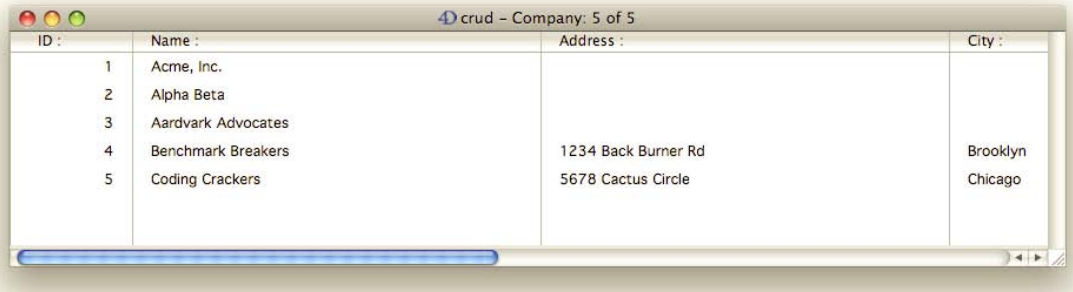
Create (作成)

この例題を実行するには、Web ブラウザーを開いて Create.php ページへの URL へアクセスします。Web サーバーと同じマシン上でブラウザーを開き、Web の公開ポート番号が 80 だとすると、URL は http://localhost/CRUD/Create.php になります。すべてが正しく設定されていれば、以下のような Web ページが表示されるはずです:



この例題では、Web ページの実際の内容は重要ではありません。私たちが見るべきは実行される PHP のコードです。ページが表示されるとき、１つのテーブルと５つのレコードが作成されます。ページが開かれた後、4D データベースを見ることで実行された内容が反映されたことを確認することができます:





このページに書かれた PHP コードを説明します:

```
$dsn = '4D:host=localhost;port=19812;charset=UTF-8';
$user = 'Administrator';
$pswd = 'test';

$db = new PDO($dsn, $user, $pswd);
```

接続変数をセットアップし、4D SQL サーバーへの接続を作成しています。

```
$create_stmt = 'CREATE TABLE IF NOT EXISTS Company( ' .
      'ID INT NOT NULL, ' .
      'Name VARCHAR(100), ' .
      'Address VARCHAR(200), ' .
      'City VARCHAR(50), ' .
      'State VARCHAR(2), ' .
      'Zip VARCHAR(200), ' .
      'created TIMESTAMP, ' .
      'status BOOLEAN, ' .
      'Notes TEXT)';

$db->exec($create_stmt);
```

これは"Company"という名前のテーブルを作成しています。PHP のシンタックスでドット (.) は文字列の結合を行います。なので最初の 10 行は 1 つの SQL ステートメントを作成し、変数$create_stmt に格納しています。次の行で実際に SQL を実行しています。

```
$id = 3;
$today_date = date("m/d/y");
$status = 1;
$notes = "Lorem ipsum dolor sit amet, consectetur adipiscing elit. Morbi tempus pharetra justo fermentum tincidunt. Integer a felis luctus augue placerat ullamcorper eu vitae velit. Nam tincidunt .....";
```

この 4 行でレコードを追加する例題に使用する変数をセットアップしています。

```
$add_stmt = "INSERT INTO Company (ID, Name) VALUES(1, 'Acme, Inc.')";
$db->exec($add_stmt);
```

これは"Company"テーブルにレコードを 1 つ挿入します。2 フィールドにデータが代入されます。

```
$add_stmt = "INSERT INTO Company (ID, Name) VALUES(" . 2 . ", 'Alpha Beta')";
$db->exec($add_stmt);
```

これは"Company"テーブルにもうひとつレコードを挿入しますが、整数値を含めるための異なる PHP シンタックスを使用しています。

```
$add_stmt = "INSERT INTO Company (ID, Name) VALUES($id, 'Aardvark Advocates')";
$db->exec($add_stmt);
```

これも"Company"テーブルにレコードを挿入しますが、整数値を含めるためにさらに異なるシンタックスを使用しています。この場合 PHP は$id 変数を解釈して変数の値 (3) を$add_stmt で使用します。

```
$add_stmt = "INSERT INTO Company (ID, Name, Address, City, State, Zip, created, status, notes) " . "VALUES
(4, 'Benchmark Breakers', '1234 Back Burner Rd', 'Brooklyn', 'NY','012345-6789', '$today_date', $status,
'$notes')";
$db->exec($add_stmt);
```

これはさらにレコードを挿入し、すべてのフィールドに値を埋めます。

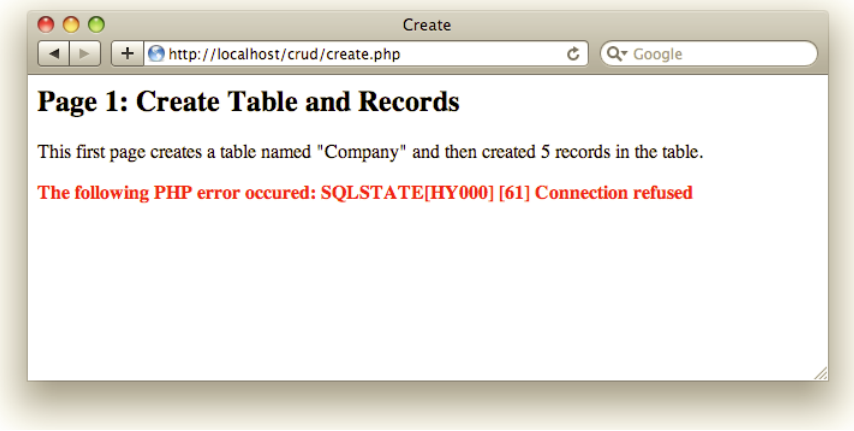
```
$add_stmt = "INSERT INTO Company " .
"VALUES (5, 'Coding Crackers', '5678 Cactus Circle', 'Chicago', 'IL', '98765-4321', '$today_date', $status,
'$notes')";
$db->exec($add_stmt);
```

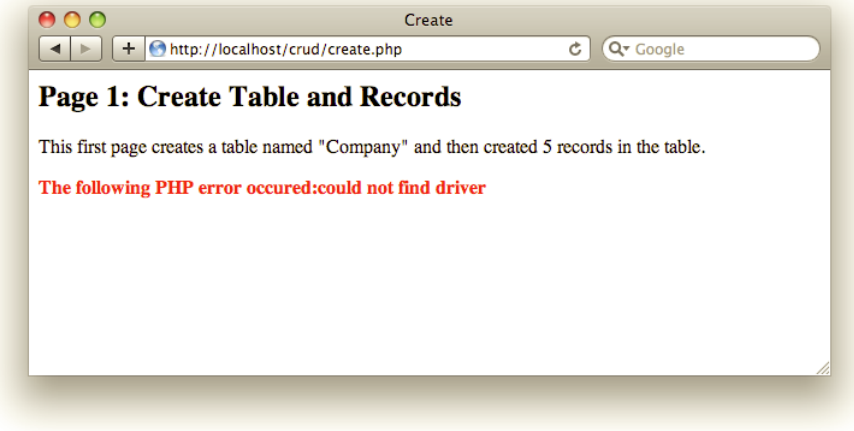
これもすべてのフィールドを埋めてレコードを挿入しますが、カラム参照を含んでいません。

最後に try/catch ブロックを含めることについて説明します。これは今回の例題のすべての 4 Web ページに含まれていて、PDO_4D ドライバーを使用する際に発生するかもしれないエラーを処理します。この例題で使用されているシンタックスは以下の通りです:

```
try {
	// do something
	// in this case, that means use PDO_4D to access a 4D database
} catch (Exception $e) {
	echo '<p style="color:red; font-weight:bold">' .
	'The following PHP error occured :' . $e->getMessage() .'</p>';
}
```

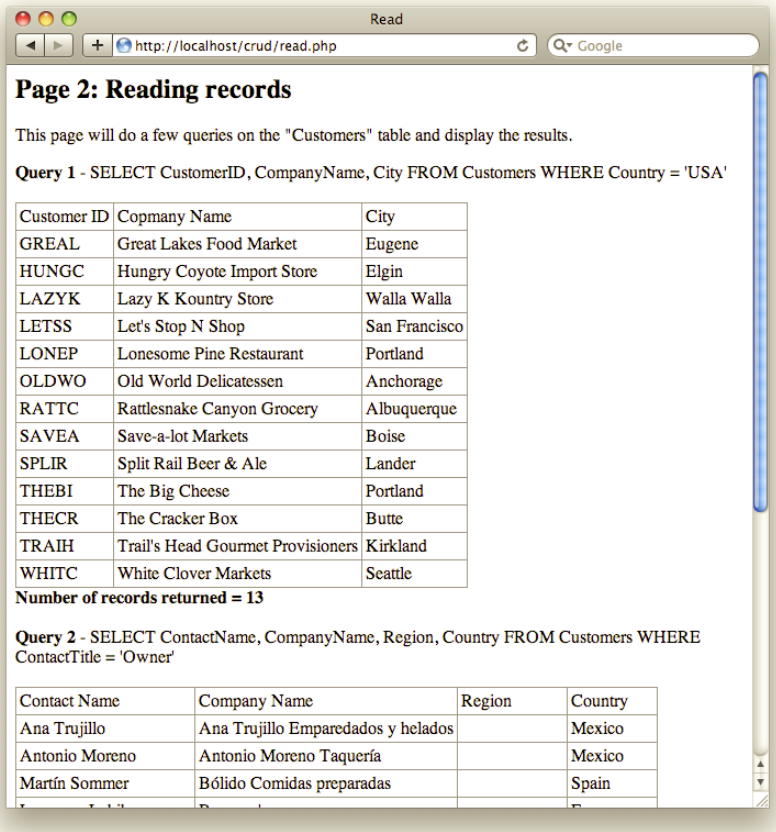
これはエラーが発生した場合に、エラーメッセージを Web ブラウザーに出力します。例えば以下は Create.php にアクセスした際に
1) ドライバーがインストールされていない
2) 4D データベースで SQL サーバーが開始されていない
ケースで受け取る可能性のあるエラーメッセージです。





Read (読み込み)

この例題を実行するには、ブラウザーを開いて Read.php ページにアクセスします。このページは 4D データベース上でクエリを実行し、結果を Web ページに出力します。このページを開くと、以下のように表示されるはずです:



このページの PHP の説明は以下の通りです:

```
$sql = 'SELECT CustomerID, CompanyName, City FROM Customers WHERE Country = \'USA\' ';
$stmt = $db->prepare($sql);
$stmt->execute();
$results_array = $stmt->fetchAll();
```

これは WHERE 句にハードコードされた値を使用した SELECT 文です。この文は先の Create の例と同様に実行しますが、今回は$results_array = $stmt->fetchAll();行が追加され、SELECT 文の結果を取得するようになっています。$results_array 内の戻り値は二次元配列に格納されています。つまりそれぞれの配列要素にはさらに配列が格納されていて、その配列には 1 レコード分のデータが収められています。

PHP では配列はさまざまに異なる構造を持つことができます。$stmt->fetchAll();関数から返される二次元配列の構造をより明確にするために、この関数から返される 2 つのレコードを例にとって記述してみます:

```
[0] => Array
    (
        [CUSTOMERID] => ALFKI
        [0] => ALFKI
        [COMPANYNAME] => Alfreds Futterkiste
        [1] => Alfreds Futterkiste
        [CITY] => Berlin
        [2] => Berlin
    )

[1] => Array
    (
        [CUSTOMERID] => ANATR
        [0] => ANATR
        [COMPANYNAME] => Ana Trujillo Emparedados y helados
        [1] => Ana Trujillo Emparedados y helados
        [CITY] => MÃ©xico D.F.
        [2] => MÃ©xico D.F.
    )
```

次に二次元配列内の個々の要素にアクセスする必要があります。

```
echo '<strong>Query 1</strong> - ' . $sql . '<br><br>';
echo '<table BORDER=1 CELLPADDING=3 CELLSPACING=1 RULES=ALL FRAME=BOX">';
echo '<tr>' . '<td>Customer ID</td>' . '<td>Copmany Name</td>' .'<td>City</td>' . '</tr>';
$index = 0;
foreach ($results_array as $id) {
    echo '<tr>';
    echo '<td>' . $results_array[$index]['CUSTOMERID'] . '</td>';
    echo '<td>' . $results_array[$index]['COMPANYNAME'] . '</td>';
    echo '<td>' . $results_array[$index]['CITY'] . '</td>';
    echo '</tr>';
    $index++;
}
echo '</table>';
echo '<br><br>';
```

このコードは返されたレコードを<table>要素内に埋め込み、HTML コードを出力しています。foreach を使用して配列の第一レベルの要素をループし、[][]シンタックスとインデックスを使用して個々のカラムにアクセスしています。この例題では適切な第一レベルの要素にインデックスとして番号を使用し、フィールド名を第二レベルのインデックスとして使用しています。先の配列構造をみると、第二レベルのインデックスとして番号を使用することができます。例えば以下のコードでも同じ結果を得ることができます:

```
…
echo '<td>' . $results_array[$index][0] . '</td>';
echo '<td>' . $results_array[$index][1] . '</td>';
echo '<td>' . $results_array[$index][2] . '</td>';
…
```

次に同じクエリの COUNT を取得しています。

```
$stmt = $db->prepare('SELECT COUNT(CustomerID) AS NumRecords FROM Customers WHERE Country =
\'USA\' ');
$stmt->execute();
$results_array = $stmt->fetchAll();
echo '<strong>Number of records returned = ' . $results_array[0][0] .'</strong><br><br>';
```

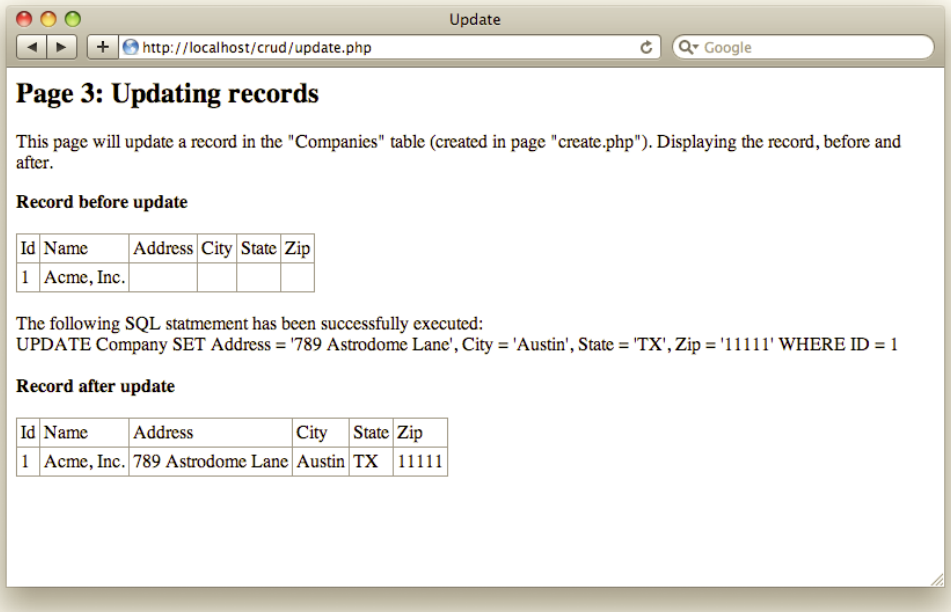
ここではクエリを実行し、$stmt->fetchAll();文で結果を取得していますが、今回は 1 つのレコードおよび 1 つのカラムしか返されないことを知っているので、[0][0]で COUNT を取得できます。

```
$title = "Owner";
$stmt = $db->prepare('SELECT ContactName, CompanyName, Region, Country FROM Customers WHERE
ContactTitle = \'' . $title . '\' ');
$stmt->execute();
$results_array = $stmt->fetchAll();
```

ここではさらに別の SELECT 文を実行していますが、今回は WHERE 句にハードコードされた値ではなく PHP の変数を使用しています。コードの他の部分はカラム名が異なるほかは、先の例と同じです。

Update (更新)

この例題を実行するには、ブラウザーを開いて Update.php ページにアクセスします。このページでは 4D データベースの 1 レコードを更新して、更新を行う前と後の結果を Web ページに出力します。ページを開くと、以下が表示されるはずです:



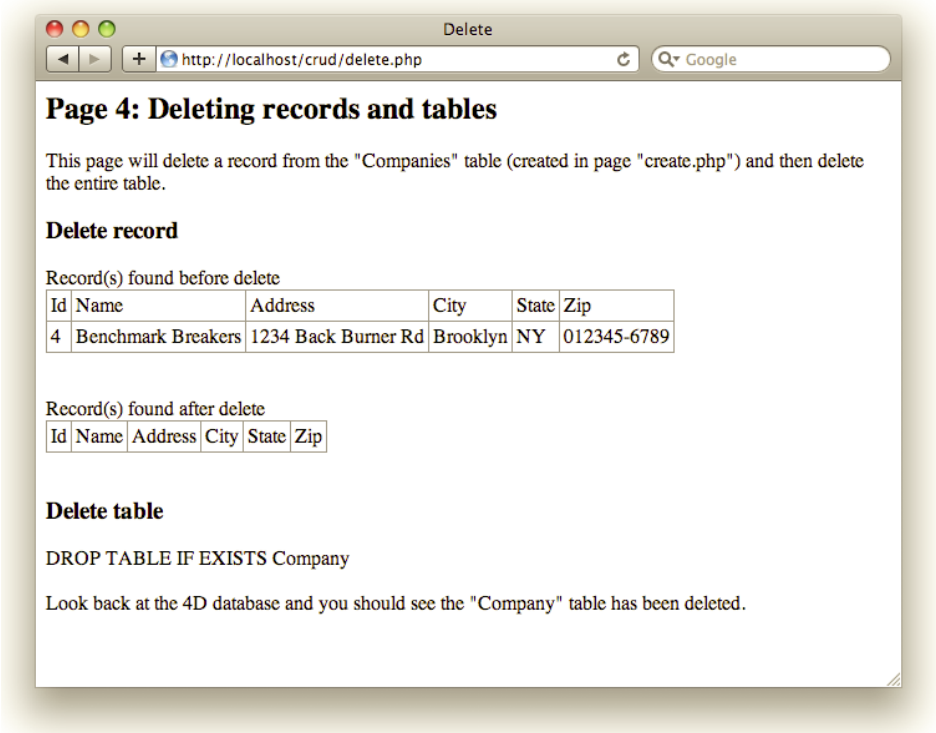
このページのコードは前の例題とほとんど同じものです。主たる新しい関数は以下の 3 行です:

```
$sql_update = 'UPDATE Company SET Address = \'789 Astrodome Lane\', City = \'Austin\', State = \'TX\', Zip = \'11111\' WHERE ID = 1 ';
$stmt = $db->prepare($sql_update);
$stmt->execute();
```

これは今までと同様のセットアップを行って、SQL の UPDATE 文を実行し、いくつかのフィールドのデータを更新しています。シングルクォート文字 (') をエスケープしている点に注意してください。

Delete (削除)

この例題を実行するには、ブラウザーを開いて Delete.php ページにアクセスします。このページは 4D データベースから 1 レコードを削除して、削除前後の結果を Web ページに出力します。このページを開くと以下のように表示されるはずです:



このページのコードは前の例題とほとんど同じものです。主たる新しい関数は以下の 3 行です:

```
$sql_delete = 'DELETE FROM Company WHERE ID = 4 ';
$stmt = $db->prepare($sql_delete);
$stmt->execute();
```

UPDATE の例題と同様、SQL 文を変更し、レコードを削除しています。

```
$sql_delete = 'DROP TABLE IF EXISTS Company ';
$stmt = $db->prepare($sql_delete);
$stmt->execute();
```

例題全体を終了するにあたり、テーブルを削除しています。このコードを実行すると、4D データベースから "Company"テーブルが削除されます。

まとめ

Apache や IIS などサードパーティーの Web サーバーで 4D をバックエンドとして使用することで、新しくエキサイティングな新しいドアが開かれます。これにより 4D の開発者はソリューションの開発時にさらなる選択肢を得ることができます。また SQL コードを使用することで、Web の開発者が 4D を知る必要が無くなります。これは PHP の

エクスパートが、データベースのことをあまり気にせずに、高品質の Web サイトの開発にフォーカスできることを意味します。

PDO_4D プロジェクトがなんであるか、またどのように PDO_4D ドライバーを使用するのかを見てきました。これで PHP をサポートするサードパーティーの Web サーバーのバックエンドとして 4D を使用するために必要なスキルを得たことになります。

その他の情報

このテクニカルノート内で議論した様々なテクノロジーに関するドキュメントへのリンクは以下の通りです:

http://pecl.php.net/package/PDO_4D/download - PDO_4D ソースのダウンロードページ
http://php.net/manual/ref.pdo-4d.php - PDO_4D ドキュメント
http://php.net/ - PHP のドキュメントと例題
http://www.apache.org/ - Apache Software Foundation ページ
http://www.iis.net/ - Microsoft Corporation IIS ページ