CHAPTER 15

MySQLを利用するためのPHP

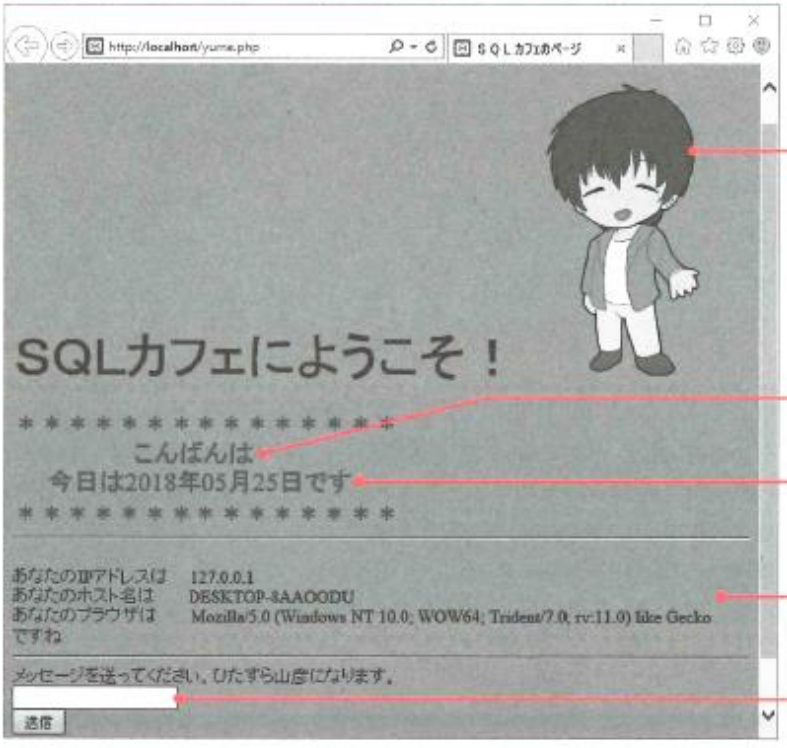
第14章まで、MySQLをCUI環境のMySQLモニタ上で実行してきました。「データベースって、こんなにも味気ないものなのか…」と思った方も多いかもしれません。

MySQLの世界はMySQLモニタだけではありません。ここからは、多くの人が日ごろ目にしているWebブラウザを使って、MySQLを操作していきます。「Apache＋MySQL＋PHP」という、最強かつ相性のよい組み合わせを駆使して、MySQLの多彩な使い方を勉強していきましょう。

第15章・16章で作成するサンプル

この章と第16章では、MySQLを制御するためのスクリプト言語「PHP」と、HTMLの基礎知識を勉強します。ここで登場するスクリプトを合わせると、次のようなWebページを作ることができます（まだMySQLは使いません）。

FIG 15-01 完成したサンプル「yume.php」



ダウンロードしたサンプルの「to_htdocs」フォルダにある「boy.gif」の画像。自由に配置してください

時間で変化するメッセージ(→P.376)

今日の日付表示(→P.363)

クライアントのIPアドレス、ホスト名、ブラウザ名表示(→P.365)

POST送信(→P.412)

サンプルを作る際には、自分が将来公開するWebページを想定しながら、自由にデザインを楽しんでみてください。

MySQLをWebアプリケーションで利用する

第14章までは、「MySQLを使う」ことを勉強してきました。MySQLモニタで「SELECT」や「INSERT」といったSQLのコマンドをキーボードで叩いて、初めてMySQLはいうことを聞いてくれました。つまり、「MySQLのデータベースを利用するなら、SQLが使えること」が前提になっていたのです。

もしMySQLを操作する方法がこれしかなかったら、世界中でこんなにMySQLが使われることはおそらくなかったでしょう。

MySQLの使い方は、そんな無味乾燥なものだけではありません。「ブラウザ上のボタンをクリックする」だけでMySQLデータベースを操作する！　そんなアプリケーションを作ることもできるのです。つまり、「ボタンをクリックすると、割り当てられた機能を実現するSQL文が発行される」という仕組みを作っておけば、使う人が難解なSQL文を知らなくても、ブラウザで簡単にデータベースが利用できるのです。実際に、現在では掲示板、ネットショップ、販売管理システム、学校の教務管理システムなど、さまざまな分野でMySQLを使ったシステムが利用されています。

このように、ブラウザをユーザーインタフェースにして、Webサーバー側に配置したMySQLなどをネットワーク越しに操作するシステムを**Webアプリケーション**といいます。Webアプリケーションを作るためには何らかのプログラム言語が必要ですが、MySQLは「Perl」「C」「PHP」「Java」など、実にたくさんのプログラム言語に対応しています。

Webを利用する仕組み

Webサーバーとクライアント

Webを利用する仕組みを考えてみましょう。Webページにはハイパーリンクが埋め込まれ、これをクリックすることで世界中のコンテンツを手に入れることができます。ハイパーリンクには「○○という場所の××という文書」といった情報が入っています。

Webサーバー

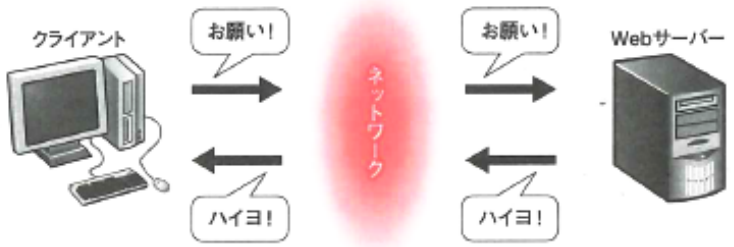
この「○○という場所」とは、特定の**Webサーバー**を意味します。Webサーバーは

インターネット上に接続されたマシンで、サーバーとしての機能を果たすためのアプリケーションなどが設定されています。サーバーは、アクセスされた場合にその要求通りに、保管しているデータを返したり、指示された処理を実行してその結果を返すなどの仕事を行います。

クライアント

Webサーバーを利用する利用者のコンピュータを、**クライアント**といいます。利用者がハイパーリンクをクリックすると、クライアントは指定した「Webサーバー」に「××を送ってよ！」という情報を送ります。この要求を受け取ったWebサーバーは、指定されたテキストや画像などのデータをクライアント、つまり皆さんのWebブラウザに向けて送り返しているのです。

FIG 15-02 Webサーバーとクライアント



Webサーバーの役目

「クライアントの要求に応えて、Webサーバーがコンテンツを送信する」という処理は、**HTTP**（HyperText Transfer Protocol）という**プロトコル**でやり取りをしています。プロトコルとは、コンピュータ同士が通信するときの手順（規則）のことです。

WebページのURLは「http://…」と書きますが、この「http://」の部分で「HTTPプロトコルで通信するよ！」と宣言しているのです。Webサーバーは、「クライアントからHTTPプロトコルに基づく要求があると、該当する文書や画像を送り返す」という機能を持ちます。HTTPのように「送ってね→送ったよ→終わったよ」とすぐに通信が終了するプロトコルを「**ステートレス**なプロトコル」といいます（→次ページCOLUMN）。

ちなみに「サーバー」（server）の「serve」（サーブ）というのは「奉仕」や「提供」という意味で、料理を給仕するスプーンやフォークも「サーバー」といいます。それから、テニスで最初に球を打つことを「サーブ」（serve）といい、「奉仕」そのものを意味する「サービス」（service）なんて言葉もありますね。

ApacheとWebサーバー

Webサーバーの機能は、インターネットにサーバーとして接続しているコンピュータ上のアプリケーションが、その処理を担当します。このアプリケーションとしてはApacheソフトウェア財団が提供しているApache（アパッチ）が有名で、世界中にあるたくさんのWebサーバーで利用されています。ApacheはMySQLと同じようにオープンソースで、無料で利用することができます。世界中の人がボランティアで、日々Apacheのプログラムを改良しています。プログラムを修正したり改良したりするためのプログラムを「patch」（パッチ）といいますが、このパッチを集めて開発したので「Apache」（アパッチ）という名前がついた、といわれています。

URL **The Apache HTTP Server Project**
http://httpd.apache.org/

MySQLや後述する「PHP」との相性では、Webサーバーアプリケーションには Apacheが最適だといえるでしょう。本書では「MAMP」を使った導入例を紹介していますが、この場合Webサーバーアプリケーションには Apacheを利用することになります。

COLUMN ▶ **ステートフルなプロトコル**

HTTPはステートレスなプロトコルですが、これに対して、たとえばWeb上でファイル転送などに用いられる「FTP」（File Transfer Protocol）のように、接続の状態を保持するものを「ステートフルなプロトコル」といいます。

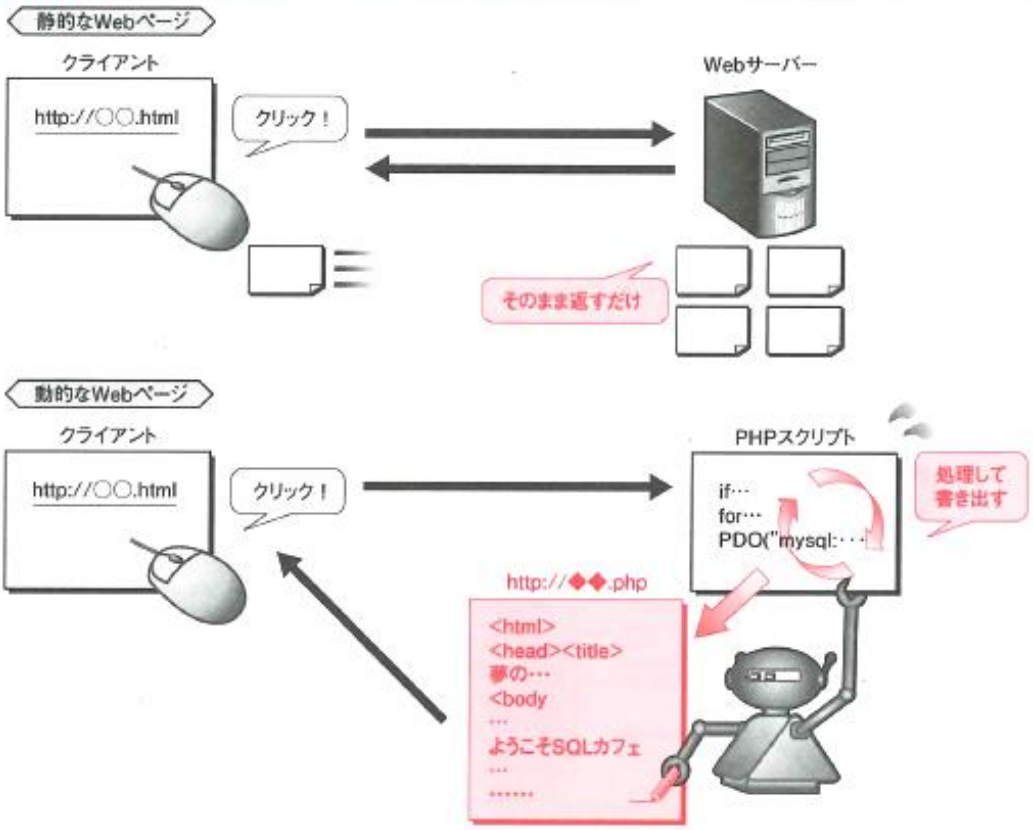
静的・動的なページ

初期のWebページは、「ハイパーリンクをクリックすると、対応するファイルが送られる」だけの単純なものでした。このような仕組みを、「静的」なページといいます。Webページの正体は、HTMLに従って記述されたテキストファイルです（→P.388）。

これに対して、最近では「動的」なWebページが多くなってきました。動的なWebページでは、クライアントから送られたデータをサーバーが処理し、それに応じたWebページをクライアント側で表示させることができます。

こうした「サーバー側の処理」の機能を実現するのが、「Perl」「Java」「PHP」…などのプログラム言語です。

FIG 15-03 静的・動的なWebページ



Web上で動作するプログラム

Web上で動作するプログラムの仕組みとしては、「CGI」と「スクリプト」の2つが有名です。

CGI（Common Gateway Interface）

プログラムをサーバーに置いて、Webブラウザからの呼び出しに応じて実行するものです。CGIプログラムを作る場合にはさまざまな言語がありますが、特に「Perl」が有名です。Perlは、古くから「アクセスカウンタ」の作成などで使われていました。

スクリプト

本来「スクリプト」は、処理を自動的に実行するために作られた、簡易的なプログラムのことを意味します。単独でテキストファイルとして作成したり、HTMLファイルの中に記述したりと、さまざまな形式のものがあります。

Webで動作するスクリプトに限っていえば、HTMLファイルの中にスクリプトの記述を含めておき、必要に応じて動作させる、といったものが一般的です。

現在、Webで利用されている一般的なスクリプトには「クライアントで動作」するものと、「Webサーバーで動作」するものの2種類があります。

クライアントサイドスクリプト

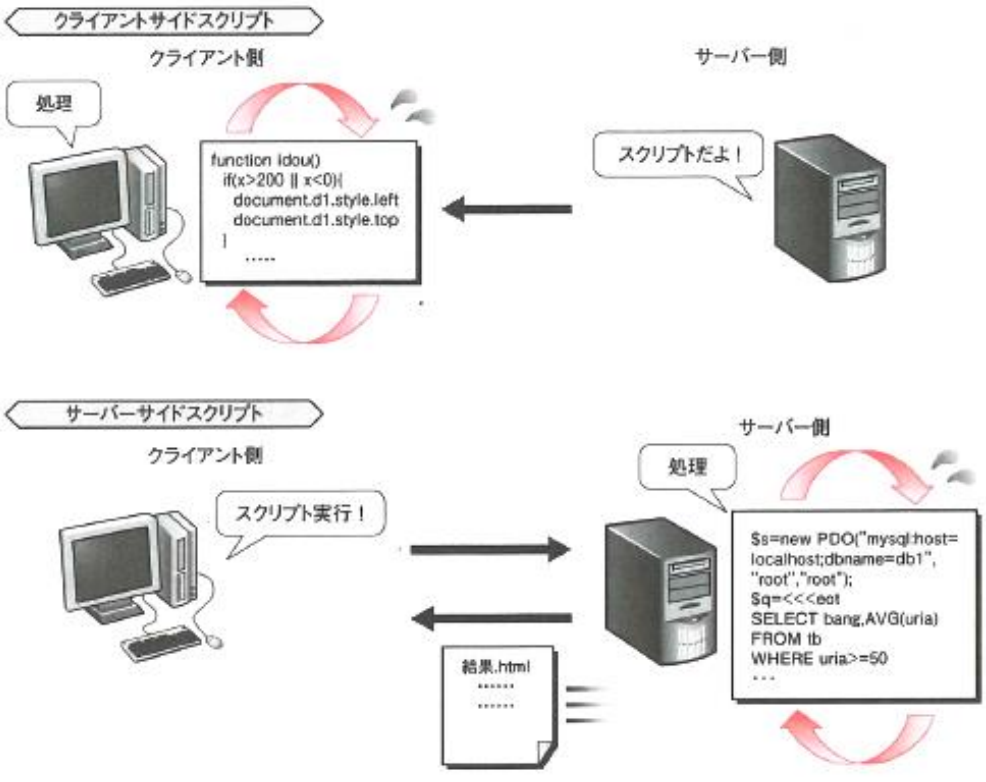
「JavaScript」などの、クライアントで動作するスクリプトです。Webサーバーとはまったく関係なしに、Webページを閲覧するパソコンの中でプログラムを実行します。つまりクライアントサイドスクリプトでは、Webサーバーはクライアント側にスクリプトを送ったら、後はどうなろうと何も手出しはできないのです。

クライアントの環境で動作するものなので、ブラウザ上の表示や動作を簡単に制御することができます。ただし、ブラウザの種類によって動作が違ったり、動作しなかったりといったことが起きることもあります。

サーバーサイドスクリプト

クライアントではなく、Webサーバーの中で実行されるのがサーバーサイドスクリプトです。クライアントからコマンドが届くと、Webサーバーの中で処理が行われ、クライアントにはその処理の結果が送られます。クライアント側は、処理された結果を見るだけです。一般的にデータベースの処理はサーバー側で行われるので、サーバーサイドスクリプトが適しています。

FIG 15-04 クライアントサイド・サーバーサイドスクリプト



PHPとは

本書では、Webアプリケーションを作成するプログラム言語として、「PHP」を利用する例を紹介します。

PHPとは何か

PHPは、Webサーバー側で動作するサーバーサイドスクリプトです。実行するためのモジュール（最小単位のプログラム）はWebサーバーにあり、これをスクリプトのコマンドによって動作させるため、一般的に動作は軽快です。

PHP : Hypertext Preprocessor
http://php.net/

PHPは正式には「PHP : Hypertext Preprocessor」という名称で、まさにWebアプリケーションを開発するためのプログラム言語です。Apacheや、Microsoft社の開発したWebサーバーアプリケーション「IIS」（Internet Information Services）など多くのWebサーバーに対応し、MySQLをはじめとした多くのRDBMSをサポートしています。そして何よりも、手軽にスクリプトが作れる親しみやすい言語なのです。

世界中の膨大な数のWebサーバー上で、PHPが利用されています。PHPは、MySQLを扱うWebアプリケーションを作るプログラム言語として、よく使われているものの1つといえるでしょう。

本書で扱うPHP

本書で紹介するPHPの知識は、MySQLをWebアプリケーションで扱うために最低限必要な基礎知識にとどめています。PHPの持つ機能は膨大で、そのすべてを本書一冊で解説することはできません。

そこで本書では、

- **第14章までで学習したSQLコマンドを、PHPを使って発行する**

という内容に限定して解説しています。

大規模な運用を前提としたデータベースサーバーを構築したり、レンタルサーバーを利用したりする手順を本の通りになぞることはできます。でも、MySQLの本質的な理解や自分で応用することにはつながりません。

以降のサンプルは、実に簡単な掲示板です。それでも、MySQLを活用する上でのエッセンスが凝縮されており、基礎として大いに役立つはずです。ぜひ1つ1つを自分

で作成し、実際に試しながら読み進めてください。

php.iniの設定

PHPの勉強をはじめる前にphp.iniの設定ファイルに対して、日付・時刻を扱うためのタイムゾーンの設定、そしてマルチバイト文字列に関する設定をしておきましょう。

php.iniはPHPの動作を設定するためのテキストファイルで、本書の環境では「C:¥MAMP¥conf¥php7.1.5」フォルダの中にあります。confフォルダ内はさまざまなPHPのバージョンごとにフォルダが分けられていますので、バージョンを間違わないようにしましょう。PHPのバージョンは、MAMPのスタートページ(→P.21)から「phpinfo」をクリックすることで確認することができます。

タイムゾーンの設定

MAMP付属のPHPでは、タイムゾーンのデフォルト値がUTCに設定されています。UTCとは世界各地の標準時を決めるときの基準となる「世界標準時」のことで、日本の時刻とは9時間ほどの時差があります。このためdate関数(→P.363)などで時刻を表示するとずれてしまうので、タイムゾーンを再設定する必要があります。

php.iniファイルをテキストエディタで開き、703行目付近にある「;date.timezone =」の先頭の「;」を削除し、値を「Asia/Tokyo」にします。

```
[Date]
; Defines the default timezone used by the date functions
date.timezone = Asia/Tokyo
```

「;」を削除して、「Asia/Tokyo」を追加

マルチバイト文字列に関する設定

本書で使用しているPHP7.1.5では、デフォルトがUTF-8に設定されているので、文字エンコーディングについては設定が不要です。これにより動的に生成された文字列は自動的にUTF-8で出力されます。

しかし、日本語を扱う場合は、マルチバイト文字列(ひらがなや漢字など2バイト以上で表現される文字列)に関する設定を行わないと、たとえば関数の戻り値に日本語が含まれる場合に文字化けする可能性があります。

php.iniの1232行目付近の「;mbstring.language = Japanese」の先頭にある「;」を削除してください。

```
[mbstring]
; language for internal character representation.
mbstring.language = Japanese
```

先頭の「;」を削除

上記の変更が終わったらphp.iniを保存し、サーバーを再起動(→P.18)します。

このほかphp.iriではさまざまな設定が可能ですが、本書の範疇を超えますのでここでは解説しません。ご興味のあるかたはPHPの専門書などを参照してください。

まずは「ようこそ!」の1行を表示させる

Apacheの動作を確認する

あらかじめ、Apacheを動作させておく必要があります。Apacheを動作させる方法は、使用している環境によって異なります。本書の第1章で解説した「MAMP」をインストールした場合は、P.18を参考にしてください。

では、Apacheが動作していることを確認しましょう。ブラウザのアドレスバーに、

```
http://localhost/MAMP
```

と入力し、Enterを押してください。

本書P.13の通りにインストール・設定した場合、「http://localhost/MAMP」にアクセスすれば、P.21でも登場した次のような画面が表示されるはずです。

FIG 15-05 MAMPのスタートページ

これから作るPHPファイルは、もちろんインターネット上に公開できるものです。でも、とりあえず最初は、Webサーバーと同じパソコンだけで利用してみます。

COLUMN ▶ **localhostのIPアドレス**

localhostのIPアドレスは、一般に「127.0.0.1」などにも割り当てられています。つまり、一般的な設定では「http://127.0.0.1/MAMP」と入力しても、前ページの画面が表示されることが多いでしょう。

まずは、PHPで「ようこそ!」と表示する

Apacheが無事に動作していることが確認できたら、今度はPHPで「こんにちは！」と表示してみることにしましょう。表示されなかったり、文字化けしたり、といろいろ問題の起こることがあるかもしれませんが、1つ1つ解決していくことにしましょう。

どのテキストエディタを使うのか?

PHPスクリプトを作成するときは、UTF-8に対応したエディタを使ってください。本書では、PHPスクリプトやHTMLの文字エンコーディングには、UTF-8を使用しています。

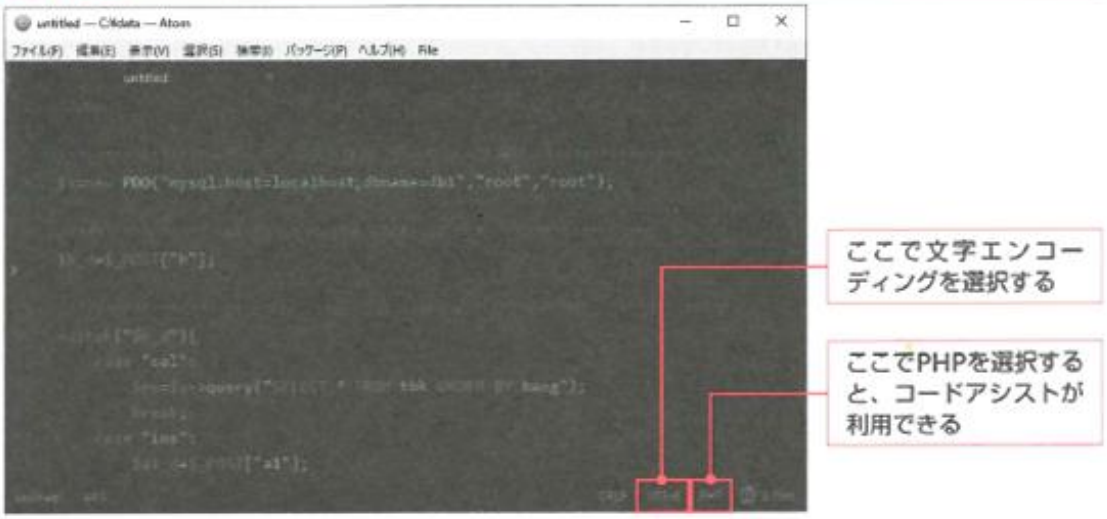
Windowsのメモ帳でも、UTF-8のテキストファイルを読み書きすることができますが、メモ帳の場合、UTF-8のファイルにBOMが付いてしまいます。BOMとは、ファイルの先頭に付くユニコード(UTF-8など)の種類を示すマークです。BOMが付くと予期せぬエラーが発生するプログラムもあるので、おすすめしません。

GitHubが開発しているAtomは、デフォルトの文字エンコーディングがUTF-8(BOMなし)ですし、PHPのプログラムを書く場合はコードアシスト機能なども使用できるので大変便利です。

Atom
https://atom.io/

オープンソースなので、無料で利用することができます。またAtomにはさまざまな追加プログラムが「パッケージ」という形態で提供されていますので、自分好みのカスタマイズが可能です。

FIG 15-06 Atomの画面



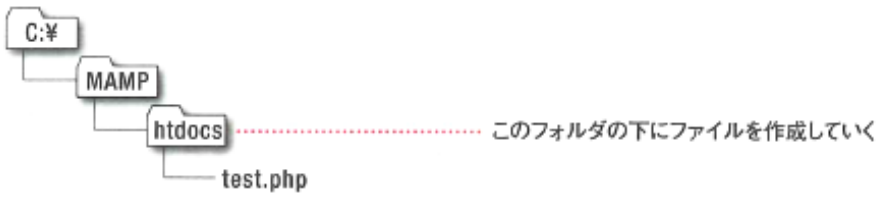
それではPHPスクリプトを作成してみます。UTF-8対応のエディタを起動してください。そして起動したら、次のように入力します。「ようこそ！」の部分だけは日本語で、それ以外は必ず半角で入力してください。

LIST 15-01 test.php

```
<?php
print "ようこそ！";
?>
```

PHPスクリプトを記述する上での文法規則に関しては、第16章で勉強します。とりあえずは、何とかPHPスクリプトが動作するようにしておきましょう。

作成したファイルを、Webサーバーによって公開されるフォルダ(→P22)に保存します。保存場所はApacheの設定により異なりますが、本書の環境ではC:¥MAMP¥htdocsフォルダになります(Windowsの場合)。したがって、これから作るPHPのファイルも、ここに保存することになります。



保存ができたら、PHPスクリプトの実行です。ブラウザのアドレス欄に次のように入力すれば、「test.php」が動作するはずです。

```
http://localhost/test.php
```

次のように「ようこそ！」と表示されれば成功です。

FIG 15-07 「test.php」が動作した例

もし、このように表示されない場合は、次項「表示されないときの対応」で問題を解決してください。問題なく表示された方は、P.347に進みましょう。

なお、たとえばあなたのマシンのIPアドレスが「192.168.1.1」であった場合、ネットワーク内の他のクライアントのブラウザに「http://192.168.1.1/test.php」と入力すると、上記の画面と同じように動作します。ただし、環境のセキュリティ状況などによっては、localhost以外からのアクセスができないこともあります。

またファイル名は、「.php」という拡張子を付けて「test.php」とします。この時、文字エンコーディングをUTF-8にして保存するのを忘れないでください。

表示されないときの対応

前項で「ようこそ！」と表示されないなど、PHPを利用していく上で、想定通りの動作にならないという事態がしばしば生じます。しかし、何の問題もないのに動作しないということはありません。必ずどこかに原因があるはずです。辛抱強く1つ1つ、その原因を探っていくことにしましょう。

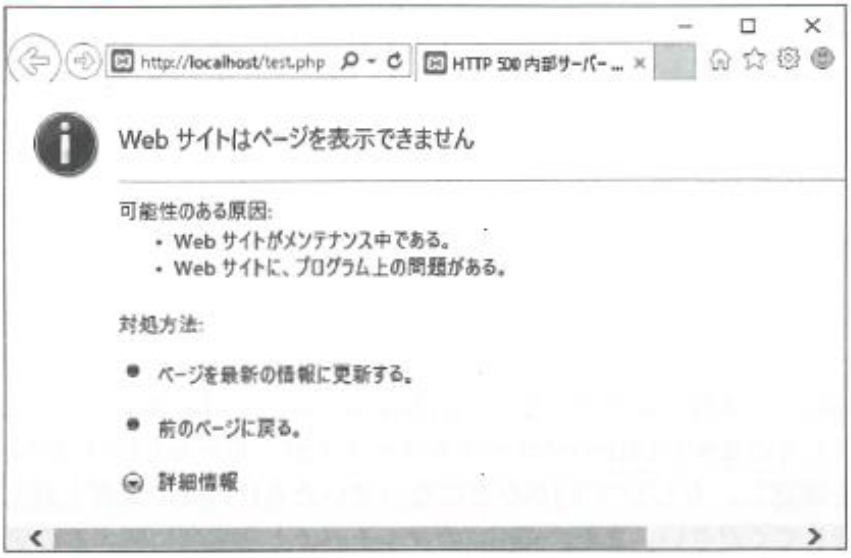
PHPがうまく動作しない場合、主に次のような原因が考えられます。

スペルミス

成功しない場合、まず一番考えられるのが、PHPファイルのスクリプト中のスペルミスです。「<php?」や「<php」「prnt」になっていたり「?>」「"」がなかった、などはよくある話です。もちろん、「ようこそ！」以外はすべて半角でなければいけません。

多くの場合、次のようなエラーメッセージが表示されます。

FIG 15-08 エラー表示画面



環境によってはエラーが発生しても、エラーメッセージが表示されないことがあります。

全角スペースが入っている

プログラムの世界では、全角スペースは「文字」として扱われます。SQL文でもそうだったように、文字は「" "」か「' '」で囲まなければいけません。つまり、インデントやキーワードの間を空けるために全角スペースを使うとエラーになってしまいます。日本語を使う限り、全角スペース問題から逃れることは難しいものです。十分気を付けましょう。

PHPが使える環境になっていない

そもそも、PHPが使える環境になっていますか？　PHPはインストールされていますか？　意外と見落としがちなミスです。第2章を読み返し、設定をもう一度確認してみましょう。

また、Webブラウザで表示させるため、Apacheが起動していることは大前提です。MAMPをインストールした場合、アドレスバーに「http://localhost/MAMP」と入力するとP.342のように表示されることをもう一度確認してください。

違うフォルダに保存している

Apacheによって公開されるフォルダは決まっています。ただし、この公開されるフォルダは、環境によって異なります。ちゃんと、正しいフォルダに保存していますか？　もう一度確認してみましょう（→P.344）。

拡張子が「.php」になっていない

公開するPHPファイルの拡張子は.phpでなければいけません。別の拡張子が付いていませんか？　必ず拡張子を表示する設定にしてから（→P.13）確認してください。テキストエディタを使っていると、ファイル名が「test.php.txt」などになり、実は拡張子が「.txt」になっていた、ということもあります。

文字化けする

本書で解説している環境であれば、基本的には文字化けは発生しないはずなのですが、万が一発生してしまったら、以下の点を確認してください。

前述のとおり、本書ではブラウザへの出力はユニコード（UTF-8）で行うことを前提として解説しています。PHPのプログラムファイルが、ちゃんとUTF-8で保存されていることを確認し、もしシフトJISなどになっていたらUTF-8で保存し直してからもう一度実行してください。また、php.iniのマルチバイト文字列に関する設定（→P.341）

を確認し、正しく設定してください。

これでも直らない場合は、MAMPをアンインストールしてから再度インストールしましょう。MAMPのアンインストールはスタートボタンから【MAMP】→【Uninstall MAMP】を選択することで行うことができます。

FIG 15-09 文字化けの例

PHPを使う

PHPスクリプトを記述する規則

ひとまず、1行だけですがブラウザに表示させることができました。それではPHPを基礎から勉強していくことにしましょう。

次は、PHPスクリプト記述するときの主な規則です。

- PHPスクリプトファイルの拡張子は「.php」
- PHPスクリプトは「<?php」で始まり「?>」で終わる
- 行の最後には「;」を付ける
- 文字列のデータは" "（ダブルクォーテーション）かまたは' '（シングルクォーテーション）で囲う。

FIG 15-10 PHPの基本構文

```
<?php
    print "ようこそ！";
?>
```

これで始まる

最後は「;」

これで終わる

それぞれ、詳しく見てみましょう。

PHPスクリプトファイルの拡張子は「.php」

PHPスクリプトを記述したファイルの拡張子は「.php」です。P.344の例では「test.php」というファイル名を付けました。、設定によっては他の拡張子に対応することもできるのですが、本書では「.php」で統一しています。

PHPスクリプトは「<?php」で始まり「?>」で終わる

「.php」の拡張子を持つテキストファイルに、次のようにスクリプトを記述していきます。

書式 PHPスクリプトの書き方

```
<?php
    実行する内容の記述;
?>
```

このうち、「<?php」の部分を**開始タグ**、「?>」の部分を**終了タグ**といいます。つまり、「<?php」があると「ここからPHPのコードが始まる」、「?>」があると「これでPHPのコードが終わる」と判断されるわけです。

P.344の例では、PHPスクリプトがファイルのすべてを占めています。でも、PHPスクリプトは、HTMLファイルの中に含めて記述することもできます。HTMLの記述に交ざっていても、「<?php」と「?>」に囲まれた部分がPHPスクリプトだとわかります。

PHPスクリプトは、HTMLの記述の中に何回でも記述することができます。HTMLタグの記述に関しては、P.391で解説します。

行の最後には「;」を付ける

行の最後には、「;」(セミコロン)を付ける必要があります。最初のうちはエラーが起こったら、まずこれを疑ってみてください。ただし、「test.php」のように1行だけでできている場合は「;」を付けなくても動作します。

文字列データは「" "」かまたは「' '」で囲む

文字列のデータは「"」か「'」で囲みます。つまり、「"」か「'」で囲まない文字列データをスクリプト中に置いてはいけない、ということです。

なお、PHPでは「" "」で囲んだ変数は展開されますが、「' '」で囲った場合は展開されません。変数を囲うときの、具体的な「" "」と「' '」の使い分けについてはP.360で解説します。

COLUMN

PHPスクリプトの記述

PHPスクリプトを記述する場合、本書では「<?php」と「?>」を使用していますが、php.iniの設定を変更することで「<?」と「?>」を使って記述できるようになります。以前は「<%」と「%>」も使用できましたが、PHP 7で使用不可になりました。

どのような処理が行われているのか

前節の「test.php」で「print "ようこそ！"」を実行すると、「ようこそ！」と表示されました。この仕組みをもう一度復習してみましょう。

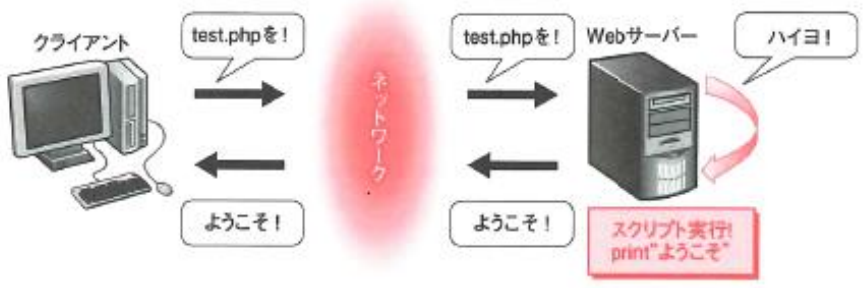
phpにアクセスする

まず、ブラウザのアドレスバーに「http://××××/test.php」と打ち込むことで、「test.php」のファイルにアクセスします。アクセスを受けたWeb サーバーは、「test.php」を実行します。

Webサーバーで処理し、結果を返す

P.344の例で実際に動作するのは、「print "ようこそ！";」の部分です。「print」は「文字列を表示しなさい」というコマンドです。つまり、このコマンドを受けてWebサーバーは「ようこそ！」の文字をクライアントのブラウザに向けて送信します。これにより、ブラウザに「ようこそ！」の文字が書き出されるのです。

FIG 15-11 行われている処理



HTMLタグを挿入する

文字列を表示するときは「print」を使いますが、HTMLタグも文字列と同じように「print」で書き出すことができます。

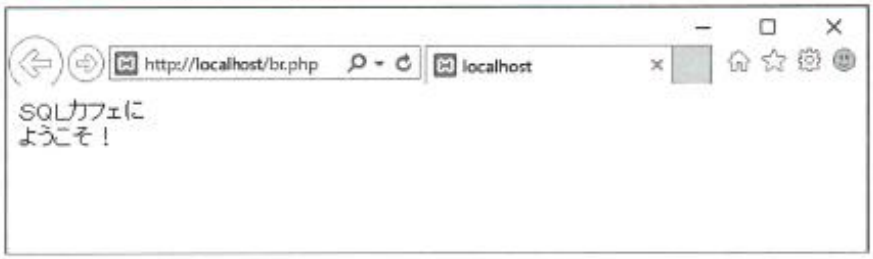
たとえば「
」は、改行を示すHTMLタグです。List15-02は、「SQLカフェに」という文字列の後に、HTMLタグの
を書き出すことで表示を改行した後、さらに「ようこそ！」と表示しています。

LIST 15-02 br.php

```
<?php
print "ＳＱＬカフェに";
print "<br>";
print "ようこそ！";
?>
```

これを実行すると、ブラウザには次のように表示されます。

FIG 15-12 「br.php」の実行例



そして、そのソースを表示させると（→P.388）、次のようになっているのが確認できます。

```
ＳＱＬカフェに<br>ようこそ！
```

COLUMN

printとecho

「print」と似た機能を持つコマンドに、「echo」があります。使い方は同じですが、「printはTRUEという値を返す」（TRUEは1として扱う）が、「echoは値を返さない」という違いがあります。

たとえば次を実行すると、ブラウザには何が表示されると思いますか？

▶print_print.php

```
<?php
print print "こんにちは";
?>
```

この場合、「こんにちは1」と表示されます。これは、右の「print "こんにちは"」が「こんにちは」を書き出して、しかも返した値であるTRUE（1）を、左のprintが書き出しているのです。

これに対して、次はエラーになります。

▶echo_echo.php

```
<?php
echo echo "こんにちは";
?>
```

これは、「何も返さないechoをechoで書き出すことはできない、echoとechoが連続しているのは誤り」なのでエラーとなった、と考えるとわかりやすいでしょう。ちなみに、同様の考えで「print echo "こんにちは";」（サンプル「print_echo.php」）はエラーなりますが、「echo print "こんにちは";」（サンプル「echo_print.php」）はエラーにはならず、「こんにちは1」が表示されます。

本書では統一して「print」を使っています。

コメントの書き方

スクリプトの途中に、自分なりのメモを書き込んでおくことができます。このメモをコメントといいます。コメントの部分は、プログラムの実行時に無視されるので、何を書いてもかまいません。

時間が経つと、プログラムを作った本人もその内容を忘れてしまいがちです。どこがどのような処理であったかという備忘録として、ポイントとなる箇所に書き込んでおくとよいでしょう。また、うまく動作しなかったとき、疑わしい記述をコメントにして動作を止めておけば、問題を発見するのも容易になります。

コメントは、次の規則で記述します。

- 「//」または「#」で始めて記述すると、その行は何も実行されない
- 「/*」から「*/」までの間に記述すると、その部分は何も実行されない

List15-03はコメントを入れた例です。これを実行しても、P.344と同じように「print"ようこそ！"」だけが実行されます。「/」や「*」を使ったデザインにして、コメントを目立たせることもできます。

LIST 15-03 comment.php

```
<?php
//  この行はすべてコメントになります。
#  この記号もコメントになります。
/*
複数行に
わたる
コメントです
*/
print "ようこそ！";                    /*  行の中にコメントを入れています  */
/**********************  目立つデザインのコメントです  **********************/
//////////////////////////  これも目立つでしょう  //////////////////////////
?>
```

phpinfo関数

今度は、PHPの関数を使ってみることにしましょう。PHPにはものすごい数の関数があります。文字列、数値から始まり、ファイル・データベース操作、ネットワーク…、この世の中に存在する処理を関数としてすべて用意したのでは？と思いたくなるほど膨大です。

関数についてはこの後の章で詳しく解説しますが、ここではまず、PHPの環境情報を表示するphpinfo関数を使ってみることにしましょう。

List15-04は、phpinfo関数を実行するPHPスクリプトファイルです。

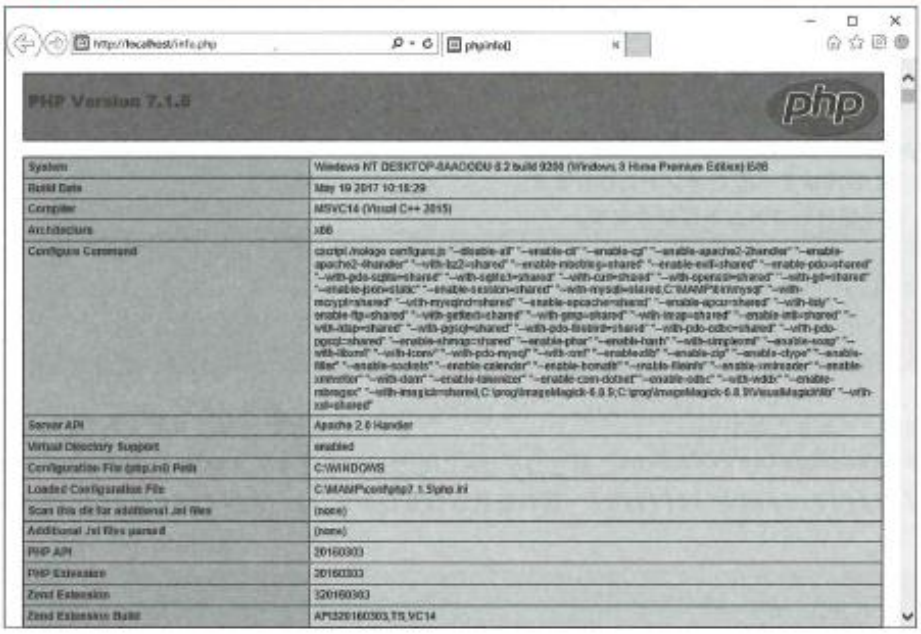
LIST 15-04 info.php

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

PHPの関数の後に記述する「( )」の中には、関数に渡す「引数」(→P.124)を書きます。この場合、もし関数が引数を取らなくても () は付けなければいけません。MySQLの関数と同じですね。

これを、「info.php」というファイル名で、公開するディレクトに保存してください。アドレスバーに「http://localhost/info.php」と入力すれば、次のように表示されるはずです。

FIG 15-13 実行結果(http://localhost/info.php)



表示されている内容を見ると、PHPのバージョンや現在の設定内容のほか、PDO(→P.422)のドライバに関する情報、Apacheにより設定される環境変数についても表示されています。本書ではその詳細には触れませんが、「REMOTE_ADDR」などの環境変数については、後で紹介します(→P.365)。

PHPやMySQLが思い通りに動作しないときは、この関数を実行して設定をチェックするとよいでしょう。なお、phpinfo関数は、MAMPのスタートページ(http://localhost/MAMP/)から、画面上部にある「phpinfo」を選択して実行することもできます。

COLUMN

PHPでシステムをシャットダウンしてみる

PHPには、実にたくさんの関数が用意されています。その中の面白いものを1つ紹介します。

コマンドを実行する関数に「exec」というものがあります。execは、引数に指定したコマンドを実行してくれます。つまり、コマンドプロンプトやターミナルで入力するコマンドの代わりになるわけです(ただし、すべてのコマンドが実行されるわけではありません)。

システムのシャットダウンは「SHUTDOWN -s」です。つまり「exec("SHUTDOWN -s ")」というPHPスクリプトを作成し、実行すると1分後にシステムが停止します。停止するまでの時間を指定する場合は、「SHUTDOWN -s -t 10」(10秒後に停止)となります。

次のPHPスクリプトをlocalhostで実行すれば、自分のパソコンが10秒後にシステムが停止します。

LIST 15-05 shutdown.php

```
<?php
exec("SHUTDOWN -s -t 10");
?>
```

なお、PHPが実行されるのはサーバー側(本書ではlocalhost)です。つまりシャットダウンされるのはサーバー側ですのでお間違えなく。

まとめ

この章では、以下のような事柄について解説しました。

- Apache、MySQL、PHPの働きとWebアプリケーション実行における役割
- PHPが利用できるようになるまでの手順
- PHPスクリプトの書き方の基本
- PHPスクリプトを動作させる方法

チェックしてみよう

この章で学んだ以下の事柄を理解しているかチェックしてみましょう。

- ❒ Webサーバーの働きを理解している
- ❒ PHPスクリプトを記述する基本的な規則を知っている
- ❒ PHPスクリプトを動作させることができる
- ❒ 文字列を書き出すPHPスクリプトを作成することができる

練習問題

問題1

次は、「指定した秒数後に、指定したURLに自動的に移動」する機能を持つタグです。これを使って、「5秒後に、http://www.softbank.co.jp/のWebページに自動的に移動する機能を持ったPHPスクリプト」を作成してください。

■<meta http-equiv='refresh'>タグの機能

```
<meta http-equiv='refresh' content='移動までの秒数;url=移動先のURL'>
```

HINT PHPスクリプトを使って、タグを書き出すだけ

解答例

問題1

次のPHPスクリプトを作成します。

LIST 15-06 サンプル「idou.php」

```
<?php
print "<meta http-equiv='refresh' content='5;
url=http://www.softbank.co.jp/'>";
print "５秒後にソフトバンクのページに移動します";
?>
```