第 5 章

PHP を用いた Web プログラミング

本章ではいよいよ Web アプリケーションを作成する。第 1 章に述べたように，Web アプリケーションのメリットは，各クライアントマシンの差異を OS とブラウザで吸収してくれるため，(Web) サーバ側で用意するスクリプト・HTML ファイルは一種類で良く，複数のアクセスにも自動的に Web サーバが対応してくれる。その分，仕組みが複雑で，トラブルになると原因を追及するのが難しいという面もある。ユーザから「Web アプリが動かない」と苦情を受けても，その原因がサーバ側にあるのかユーザ側にあるのか，途中経路のネットワークにあるのかは詳細に調べないと分からない。本章以降では次第に複雑な Web アプリケーションを作成していくことになるが，もしトラブルに遭遇したら，その原因追及も勉強のうちと心得て乗り越えて頂きたい。

プログラミング学習の最終目標は，自分でコードのミスを発見できるようになること

である。

5.1 PHP スクリプトの基礎

Web アプリケーションを作成するために近年よく使用される PHP スクリプト言語に触れておこう。まず最初にコマンドラインから動作するスクリプト (ソースコード) を作成してみよう。

5.1.1 hellow.php の作成

まず，画面 (標準出力先) に”Hellow, PHP!”という文字列を表示する PHP スクリプト”hellow.php”を作成し，実行してみる。PHP スクリプトは HTML ファイルに埋め込んで使うことが多いため，本書では必ず PHP スクリプトを<?php ... ?>という文字列で囲むことにする。

hellow.php は次のようになる。

```
1:<?php
2:    echo "Hellow, PHP!¥n";
3:?>
```

echo 関数は文字列を表示する PHP の標準関数である。C と同様に

```
2:    printf("Hellow, PHP!¥n");
```

と書いても良い。

実行する際には PHP インタプリタ (php) をコマンドラインから指定して実行する。

```
$ php hellow.php
Hellow, PHP! ←文字列が表示される。
```

5.1.2　quadratic_eq.php の作成

次に 2 次方程式を入力して判別式を計算し，実数解の時には解を計算して表示，複素数解の時には”Complex!”と表示する PHP スクリプトを “quadratic_eq.php”という名前で作成する。内容は下記の通りである。

```
 1: <?php
 2:     echo "a * x^2 + b * x + c = 0¥n";
 3:
 4:     echo "a = "; $a = trim(fgets(STDIN));
 5:     echo "b = "; $b = trim(fgets(STDIN));
 6:     echo "c = "; $c = trim(fgets(STDIN));
 7:
 8:     printf("%f * x^2 + %f * x + %f = 0¥n", $a, $b, $c);
 9:
10:     $d = $b * $b - 4.0 * $a * $c;
11:
12:     if($d >= 0.0)
13:     {
14:         echo "Real solutions:¥n";
15:         printf("x1 = %f¥n", (-$b + sqrt($d)) / (2.0 * $a));
16:         printf("x2 = %f¥n", (-$b - sqrt($d)) / (2.0 * $a));
17:     }
18:     else
19:     {
20:         echo "Complex!¥n";
21:     }
22: ?>
```

プログラムの挙動は pp.36 の条件分岐を加えた C プログラム (quadratic_eq.c) と全く同じである。

この PHP スクリプトはコマンドライン版 (CUI) の PHP インタプリタ (php コマンド) に解釈されて実行される。その手順は下記の通り。

```
$ php quadratic_eq.php ← quadratic_eq.phpスクリプトを実行
a * x^2 + b * x + c = 0
a = 1 ←1を入力
b = 2 ←2を入力
c = 3 ←3を入力
1.000000 * x^2 + 2.000000 * x + 3.000000 = 0
Complex!
```

C プログラムと比較してみれば分かる通り，文法的には共通点が多い (セミコロンで一区切り，if 条件分岐書式は同じ，演算子も共通・・・等）よく似ているが，下記のような違いがある。

■変数型宣言が原則として不要　厳格にセキュリティを確保しなければならない場合は別として，必要なところで必要な変数を勝手に使用して良い。例えば

```
$a = 3;
$b = 4;

print '$a + $b = ' . ($a + $b) . "¥n";
```

のように，変数$a, $b を宣言なしで使っても単独の PHP スクリプトとして何ら問題なく動作する。

■変数は"$変数名"と表記　上記に示した通り，変数には$マークが接頭詞として付く。配列は 配列名 [0], 配列 [1], ...，あるいは添え字に文字列を指定して 配列名 [文字列 1], 配列名 [文字列 2], ... というハッシュ (hash) も使用することが出来る。

■シングルクォート (´) とダブルクォート (¨) の相違点　例えば PHP スクリプトの中で

```
print '$a + $b = ' . ($a + $b) . "¥n";
```

と書くと，$a = 3, $b = 4 の場合，

```
$a + $b = 7
```

と標準出力される。それに対して

```
print "$a + $b = " . ($a + $b) . "¥n";
```

と書くと

```
3 + 4 = 7
```

と表示される。つまり，

シングルクォート (´) でくくった文字列　・・・文字列がそのまま表示される
ダブルクォート (″) でくくった文字列　・・・文字列内の変数が展開され，変数に格納されている値が表示される

という違いがある。変数を含まない文字列はどちらを使ってくくっても構わないが，変数を含む場合は異なるので注意すること。

■文字列の結合　二つの文字列を繋げるときには，「文字列 1 . 文字列 2」のようにドット (.) で行う。

課題 A

以上の解説を理解し，次の機能を順次実装せよ。方法は C プログラムの課題 (pp.37) と同じなのでそちらを参照しながら PHP スクリプトに実装せよ。

1. 実数解を計算して表示する。この時，次の 2 つの方程式の解が正しく計算できていることを確認せよ。
   (a) $2x^2 + 12x + 18 = 0$ ($x_1 = x_2 = -3$)
   (b) $32x^2 - 732160x - 516544800 = 0$ ($x_1 = 23565,\ x_2 = -685$)
2. 複素数解であることを判別するだけでなく，実数部，虚数部の計算をそれぞれ行って次のように表示できるよう，“quadratic_eq.php”を改良せよ。

3*. $a = 0$ の時も正しく 1 次方程式 $bx + c = 0$ の解を計算して表示出来るようにせよ。

4*. どんな a, b, c の値が入っても正しく処理して計算，あるいはエラー表示が出るようにせよ。入力された文字列が数として正しい表記になっているかどうかは is_numeric 関数を使って確認できる。

5.2　Web プログラミングの基礎

第 1 章のおさらいも兼ねて，Web アプリケーションの動作システムを確認し，必要最小限の HTML の機能を説明し，HTML ファイルによる静的コンテンツの作成を行う。

5.2.1　Web システムの概要

Web アプリケーションの仕組みは pp.7 の図 1.7 に示した。ここではそのうち，ネットワークを介したブラウザと Web サーバとの間のやりとりを詳しく見ていこう。

例えば，インターネットに接続されたクライアントマシンから，`http://www.sist.ac.jp/~tkouya/index.html` にアクセスしようとする。その時，図 5.1 のようなやりとりがブラウザと Web サーバとの間でなされる。

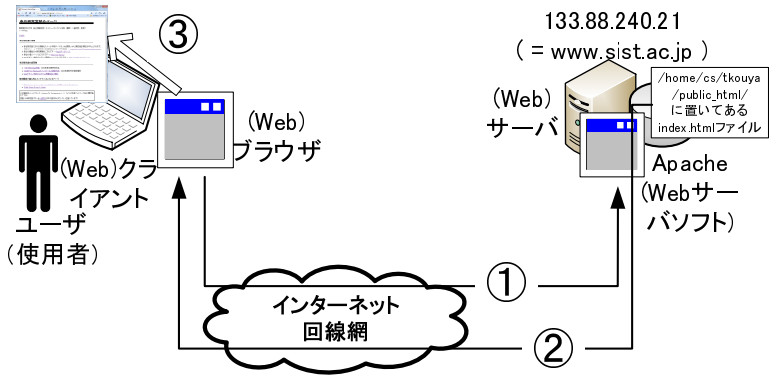


図 5.1　Web システムの構成と動作の仕組み

① Web サーバにアクセスし，GET メソッドでコンテンツ (この場合は index.html) を読み出す。

② 読み出された index.html の内容はブラウザ側に標準出力される。

③ ブラウザは読み取った index.html の内容を解釈し，必要に応じて追加の CSS ファイル，画像ファイル，埋め込みコンテンツ (Flash 動画など) を読み出して追加し，ユーザに対して画面表示する。

①における URL がどのように解釈されるかを示したのが図 5.2 である。

クライアント側で解釈されるのは左側の `http://www.sist.ac.jp` までである。最初の `http:` でアクセスする先が HTTP(HyperText Transport Protocol)，即ち Web サービスのための通信を行うということが宣言され，`www.sist.ac.jp` という FQDN(Fully Quolified Domain Name) を持つインターネット上のマシンが `133.88.240.21` という IP アドレス

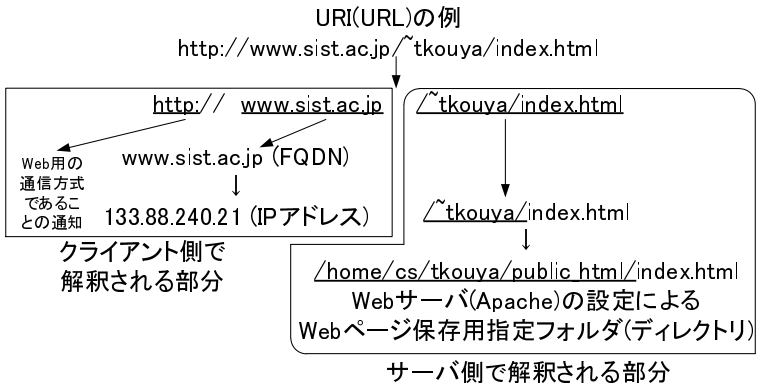


図 5.2 URL(URI) の解釈

であることを DNS サーバ (Domain Name Service) に教えてもらい，インターネットを介してこの Web サーバマシンの TCP 80 番ポートにアクセスする。

サーバ側にはこの際

```
GET /~tkouya/index.html
```

という文字列が届けられ，これによって GET メソッド (方式) により，URL(の一部) に記されたフルパス名のファイルの内容が読み出されて標準出力され，そのままクライアントマシンに送り届けられる。

5.2.2 HTML と CSS

HTML(HyperText Markup Language) は，静的 (動かない) な Web ページを作るために用いられる。**タグ** (tag) と呼ばれる<**タグ名**>・・・</**タグ名**>という括りを使って文書の構造を表現するという特徴がある。

ごく基本的な HTML は以下のような構造になる。

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8" />←Webページの文字コードを"UTF-8"に指定
    <link rel=stylesheet" href="hogehoge.css" type="text/css" />
    <title>Webページタイトル</title>
  </head>
  <body>
    ・・・Webページ本文・・・・
  </body>
</html>
```

まず外側に html タグがあり，その中に，head タグと body タグがある。head タグ内にはタイトルやこの HTML ファイルが使っている文字コード等を指定し，ブラウザ（ホームページ表示ソフトウェアの総称) 側に情報を伝える役割を果たす。body タグ内には，ブラウザで表示される本文の内容を記述する。Web サーバは自分の外部記憶装置に置いてあるこのような HTML ファイルを，外部からの接続要求に従って適宜取り出し，標準出

力してブラウザに渡しているだけである。ブラウザは HTML に記述された文書構造と CSS(この場合は hogehoge.css ファイルに記述されている) に指定された装飾指定に則って画面表示を行っている。

5.3　HTML と PHP スクリプトの配置

Web サーバを介して公開するコンテンツは全て指定ディレクトリ (この環境ではホームディレクトリ直下の public_html の下に置いておくこと。 また，必ず以下のように，ホームディレクトリと public_html ディレクトリのパーミッションを確認し，Others に分類されるユーザがアクセス可能になっていることをチェックする。

```
$ pwd
/home/tkouya/public_html
$ ls -ld ./
drwxr-xr-x 9 tkouya tkouya 4096  1月  4 20:36 ./
$ ls -ld ../
drwxr-xr-x 31 tkouya tkouya 4096  1月 20 15:09 ../
```

もしパーミッション変更が必要であれば，パーミッションの変更手順 (2.3.3, P.15) を参考にして変更しておくこと。

ディレクトリのパーミッションが確認できたら，public_html に index.html を下記の形式で作成しておく。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>幸谷智紀の制作物</title>
    </head>
<body>
<h1>幸谷智紀の制作物</h1>
<p>最終更新日: 2018-06-01 (Fri)</p>
<hr>
<ol>
    <li><a href="keisan.php">keisan.php</a></li>
    <li>2次方程式</li>
</ol>
<hr>
<address>Copyright (c) Tomonori kouya</address>
</body>
</html>
```

リモートログインしているマシンのブラウザ (Internet Explorer, Firefox, Chrome 等) から Web サーバにアクセスし，この HTML ファイルが正しく表示されていることを確認しよう。URL は

```
http://Webサーバアドレス/~ユーザ名/index.html
```

あるいは

```
http://Webサーバアドレス/~ユーザ名/
```

となる[*1]。ファイル名を省略しても良いのは，Web サーバが指定したインデックスファイル名 (index.html) を使っているからである。ファイル名が省略されたアクセスがあった時，Web サーバは指定ディレクトリに存在するインデックスファイルを探して表示する。

制作物が出来たら
〜
に箇条書き
〜
を追加し，随時制作物へのリンクを張っておく。こうしておけばどこまで完成できたか一目瞭然である。

図 5.3 index.html のブラウザ表示画面例

最後に，Web サーバから PHP スクリプトが実行可能になっているかどうかを，

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

という 3 行 PHP スクリプト (phpinfo.php) を作成し，index.html と同じディレクトリに配置して

```
http://Webサーバアドレス/~ユーザ名/phpinfo.php
```

と呼び出して実行してみよう。図 5.4 のように PHP のバージョン番号・環境設定情報が表示されれば，PHP スクリプトを Web サーバ経由で実行可能であることが確認できる。

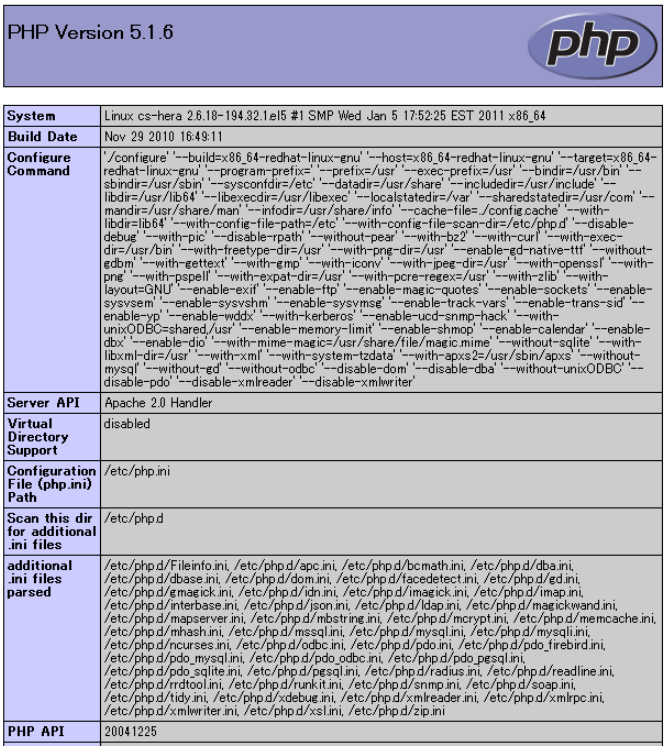
5.4 HTML フォームと PHP スクリプトとの連携

HTML ファイルでユーザからの入力を受け付けるタグのセットをフォーム (form) と呼び，次のような形式で指定する。フォームからデータを取得する方法 (というより Web でのデータ取得方法) には GET メソッド (URL にデータが反映される) と POST メソッド (URL にはデータが表示されず，環境変数を通じてデータが渡される) がある。この二つを場合に応じて使い分ける。

フォームタグは次のような形で使われるのが普通である。

```
<form action="実行スクリプト名" method="GETあるいはPOSTを指定">
```

[*1] ホームディレクトリ以下に置かれた各ユーザの Web ページの URL は，Web サーバの設定に依存しているので管理者に確認しておこう。



PHP Version 5.1.6

System	Linux cs-hera 2.6.18-194.32.1.el5 #1 SMP Wed Jan 5 17:52:25 EST 2011 x86_64
Build Date	Nov 29 2010 16:49:11
Configure Command	'./configure' '--build=x86_64-redhat-linux-gnu' '--host=x86_64-redhat-linux-gnu' '--target=x86_64-redhat-linux-gnu' '--program-prefix=' '--prefix=/usr' '--exec-prefix=/usr' '--bindir=/usr/bin' '--sbindir=/usr/sbin' '--sysconfdir=/etc' '--datadir=/usr/share' '--includedir=/usr/include' '--libdir=/usr/lib64' '--libexecdir=/usr/libexec' '--localstatedir=/var' '--sharedstatedir=/usr/com' '--mandir=/usr/share/man' '--infodir=/usr/share/info' '--cache-file=../config.cache' '--with-libdir=lib64' '--with-config-file-path=/etc' '--with-config-file-scan-dir=/etc/php.d' '--disable-debug' '--with-pic' '--disable-rpath' '--without-pear' '--with-bz2' '--with-curl' '--with-exec-dir=/usr/bin' '--with-freetype-dir=/usr' '--with-png-dir=/usr' '--enable-gd-native-ttf' '--without-gdbm' '--with-gettext' '--with-gmp' '--with-iconv' '--with-jpeg-dir=/usr' '--with-openssl' '--with-png' '--with-pspell' '--with-expat-dir=/usr' '--with-pcre-regex=/usr' '--with-zlib' '--with-layout=GNU' '--enable-exif' '--enable-ftp' '--enable-magic-quotes' '--enable-sockets' '--enable-sysvsem' '--enable-sysvshm' '--enable-sysvmsg' '--enable-track-vars' '--enable-trans-sid' '--enable-yp' '--enable-wddx' '--with-kerberos' '--enable-ucd-snmp-hack' '--with-unixODBC=shared,/usr' '--enable-memory-limit' '--enable-shmop' '--enable-calendar' '--enable-dbx' '--enable-dio' '--with-mime-magic=/usr/share/file/magic.mime' '--without-sqlite' '--with-libxml-dir=/usr' '--with-xml' '--with-system-tzdata' '--with-apxs2=/usr/sbin/apxs' '--without-mysql' '--without-gd' '--without-odbc' '--disable-dom' '--disable-dba' '--without-unixODBC' '--disable-pdo' '--disable-xmlreader' '--disable-xmlwriter'
Server API	Apache 2.0 Handler
Virtual Directory Support	disabled
Configuration File (php.ini) Path	/etc/php.ini
Scan this dir for additional .ini files	/etc/php.d
additional .ini files parsed	/etc/php.d/Fileinfo.ini, /etc/php.d/apc.ini, /etc/php.d/bcmath.ini, /etc/php.d/dba.ini, /etc/php.d/dbase.ini, /etc/php.d/dom.ini, /etc/php.d/facedetect.ini, /etc/php.d/gd.ini, /etc/php.d/gmagick.ini, /etc/php.d/idn.ini, /etc/php.d/imagick.ini, /etc/php.d/imap.ini, /etc/php.d/interbase.ini, /etc/php.d/json.ini, /etc/php.d/ldap.ini, /etc/php.d/magickwand.ini, /etc/php.d/mapserver.ini, /etc/php.d/mbstring.ini, /etc/php.d/mcrypt.ini, /etc/php.d/memcache.ini, /etc/php.d/mhash.ini, /etc/php.d/mssql.ini, /etc/php.d/mysql.ini, /etc/php.d/mysqli.ini, /etc/php.d/ncurses.ini, /etc/php.d/odbc.ini, /etc/php.d/pdo.ini, /etc/php.d/pdo_firebird.ini, /etc/php.d/pdo_mysql.ini, /etc/php.d/pdo_odbc.ini, /etc/php.d/pdo_pgsql.ini, /etc/php.d/pdo_sqlite.ini, /etc/php.d/pgsql.ini, /etc/php.d/radius.ini, /etc/php.d/readline.ini, /etc/php.d/rrdtool.ini, /etc/php.d/runkit.ini, /etc/php.d/snmp.ini, /etc/php.d/soap.ini, /etc/php.d/tidy.ini, /etc/php.d/xdebug.ini, /etc/php.d/xmlreader.ini, /etc/php.d/xmlrpc.ini, /etc/php.d/xmlwriter.ini, /etc/php.d/xsl.ini, /etc/php.d/zip.ini
PHP API	20041225

図 5.4　phpinfo.php の実行画面例

```
    <input type="text" name="a" />
    <input type="text" name="b" />
    <input type="text" name="c" />
    ・・・・
    <input type="submit" />
</form>
```

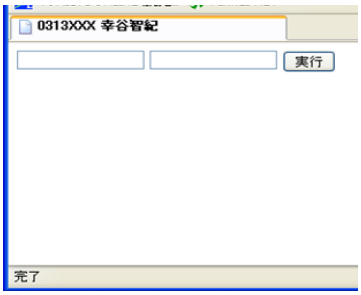


図 5.5　フォームの例

ユーザはブラウザ画面を通じてこのフォームにデータを入力する。Submit ボタンが押された段階でそのデータは Web サーバ (本章では Apache を想定) を通じて action に指定された CGI スクリプトに，”QUERY_STRING”という環境変数 (PHP では

$ENV['QUERY_STRING'] 変数でアクセスする) に格納されて渡される。この時，CGI に渡されるデータは全て連結された文字列になっている。例えば上記のフォームの a, b, c というラベル (name 属性で指定) に”32”, ”32”, ”23”という文字列が入力されたものとすると

```
a=32&b=32&c=23
```

つまり，「ラベル=入力文字列」という文字列を”&”で連結したものになっている訳である。従って，実行スクリプト側では“&”で分割し，”a=32”，“b=32”，“c=23”とし，さらに”=“以降の値のみ”32”，“32”，“23”を取得してようやく使えるようになる。

PHP スクリプトでは，Web サーバ (Apache) と連動して動作する PHP インタプリタがこの作業を自動的に行ってくれる。従って，PHP スクリプト側では，

GET メソッドを指定 → $_GET['nameプロパティ指定文字列']
POST メソッドを指定 → $_POST['nameプロパティ指定文字列']

という配列にアクセスすればよい。この点，C や Perl 等，Web サーバとの連動機能のないコンピュータ言語に比べて格段に Web アプリケーションが作りやすくなっている。

5.4.1 quadratic_eq.html ファイルの作成

ここでは 2 次方程式を解く Web アプリケーションを作成していく。そのためには係数 *a*, *b*, *c* を入力値として受け取らばならない。それをフォームで実現したのが以下の quadratic_eq.html である (body タグの中のみ記述。以下同じ)。

```
<h1>2次方程式の入力フォーム</h1>
<form action="quadratic_eq_solve.php" method="get">
      <input type="text" name="a" /> * x^2
    + <input type="text" name="b" /> * x
    + <input type="text" name="c" /> = 0 <br />
    <input type="submit" value="2次方程式を解く" />
    <input type="reset" value="係数を消去" />
</form>
```

Web サーバにアクセスし，このフォームが正しく表示されていることを確認しよう (図 5.6 左参照)。

5.4.2 action に PHP スクリプトを指定

Web アプリケーションとして動作する PHP スクリプトは以下のように，HTML ファイルの中に埋め込んで使用する。

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8" />
        <title>Solutions of Quadratic Equation</title>
    </head>
<body>
<h1>2次方程式の解</h1>
<?php
```

```
・・・・・・・・・・・・・・・・・・・・・・・・
・・・・・ここにPHPスクリプトを書く ・・・・・
・・・・・・・・・・・・・・・・・・・・・・・・
```

```
?>
<p><a href="quadratic_eq.html">戻る</a></p>
</body>
</html>
```

一つの HTML ファイル内には複数の PHP スクリプトの記述<?php ・・・ ?>が分散配置されていても構わないが，一連の PHP スクリプトとして解釈される。ファイル名は拡張子として “.php”を使うことはコマンドライン環境と変わらない。

5.4.3　HTML と PHP スクリプトの連携

最後に，quadratic_eq.html フォームの action プロパティで指定した PHP スクリプト，quadratic_eq_solve.php を作成する。前述したように HTML タグを指定し，「PHP スクリプトを書く」部分に quadratic_eq.php の処理内容を記述して，quadratic_eq_solve.php ファイルを作成すればよい。

その際，計数を入力する部分

```
4:      echo "a = "; $a = trim(fgets(STDIN));
5:      echo "b = "; $b = trim(fgets(STDIN));
6:      echo "c = "; $c = trim(fgets(STDIN));
```

は，フォームの input タグから受け取るように

```
10:     $a = $_GET['a'];
11:     $b = $_GET['b'];
12:     $c = $_GET['c'];
```

とする。

以上の変更を行って，qudratic_eq.html にブラウザからアクセスし，計数を入力して図 5.6 のように解が表示されれば O.K. である。

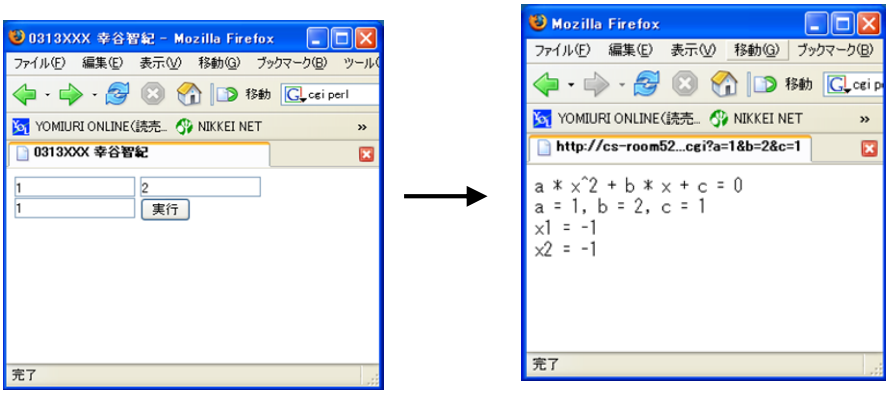


図 5.6　2 次方程式を解くフォームと表示例

以上でフォームからデータを受け取って加工する HTML, PHP スクリプトが完成したことになる。

課題 B

課題 A と同じ内容を，フォーム +PHP スクリプトからも実行できるように改変し，実行結果を確認せよ。

5.5 PHP スクリプトと HTML フォームとの一体化

入力のためのフォームと，入力データの加工を行うスクリプトは必ずしも分離していなければならない，というものではない。簡単な処理ならば一連の HTML + PHP スクリプト埋め込みで一つのファイルとしてまとまっていることが望ましいこともある。

ここで，先の 2 次方程式の例を一つの PHP ファイル (quadratic_eq_single.php) にまとめてみよう。その構造は次のようになる。

```
<h1>2次方程式の解(フォーム + PHP)</h1>
<form method="get">
  ・・・・・・・・・・・・・・
  ・・・・フォーム本体・・・・
  ・・・・・・・・・・・・・・
</form>
<?php
    // 値がフォームから入力されているかをチェック
    if(isset($_GET['a']) && isset($_GET['b']) && isset($_GET['c']))
    {
        ・・・・・・・・・・・・・・・・・・・・・・・・・
        ・・・・・2次方程式を解くPHPスクリプト・・・・・
        ・・・・・・・・・・・・・・・・・・・・・・・・・
    }
?>
<p><a href="index.html">トップに戻る</a></p>
```

フォームの action プロパティを省略すると自分自身 (このケースでは quadratic_eq_single.php) が submit ボタンが押されると呼び出される。間違えて別のスクリプトを呼び出すこともなくなる。

但し，方程式を解く際には，必ず係数が入力されている必要がある。そのチェックを行うために isset 関数を使用し，フォームから値が入力されているかどうかを確認し，入っている時のみ方程式の解を計算するようにする。

課題 C

quadratic_eq_single.php スクリプトを作成し，正確に動作することを確認せよ。

5.6 PHP スクリプト vs. C プログラム

本章の最後に，C プログラムと PHP スクリプトの動作速度の違いを体感してみよう。以下はコマンドラインから動作する行列・ベクトル積ベンチマークスクリプトである。

```
1: <?php
2:     // check dimension
3:     if($argc <= 1)
```

```
 4:     {
 5:         echo "Usage: " . $argv[0] . " [dimension] ¥n";
 6:         return;
 7:     }
 8:
 9:     // input dimension
10:     $dim = $argv[1];
11:     printf("Dimension = %d¥n", $dim);
12:
13:     // initialize
14:     $mat_a = array($dim, $dim);
15:     $vec_b = array($dim);
16:     $vec_c = array($dim);
17:
18:     // mat_a[i][j] = i + j + 1
19:     for($i = 0; $i < $dim; $i++)
20:     {
21:         for($j = 0; $j < $dim; $j++)
22:             $mat_a[$i][$j]= (double)($i + $j + 1);
23:     }
24:
25:     // vec_b[i] = dim - i
26:     for($i = 0; $i < $dim; $i++)
27:         $vec_b[$i] = (double)($dim - $i);
28:
29:     // vec_c := mat_a * vec_b
30:     $stime = microtime(true); // float型として秒数を返す
31:     for($i = 0; $i < $dim; $i++)
32:     {
33:         $vec_c[$i] = 0.0;
34:         for($j = 0; $j < $dim; $j++)
35:             $vec_c[$i] += $mat_a[$i][$j] * $vec_b[$j];
36:     }
37:     $etime = microtime(true);
38:
39:
40:     // print vec_c
41:     for($i = 0; $i < $dim; $i++)
42:         printf("vec_c[%d] = %f¥n", $i, $vec_c[$i]);
43:
44:
45:     printf("Execution time: %f [seconds]¥n", $etime - $stime);
46:
47:     // free
48:     unset($mat_a);
49:     unset($vec_b);
50:     unset($vec_c);
51: ?>
```

この動作速度を比較してみると，C 言語の場合 (matmul.c)

```
$ ./matmul 100
Dimension = 100
Clock number per second: 100 (clocks/sec)
Run Time (Clock)    : 0
Run Time (Second)   : 0.000000
System Time (Second): 0.000000
User Time (Second)  : 0.000000
$ ./matmul 200
Dimension = 200
Clock number per second: 100 (clocks/sec)
Run Time (Clock)    : 0
Run Time (Second)   : 0.000000
System Time (Second): 0.000000
User Time (Second)  : 0.000000
```

と時間計測できないほど短時間で終了している[*2]のに対し，PHP スクリプトでは

```
$ php matmul.php 100
(略)
Execution time: 0.004659 [seconds]
$ php matmul.php 200
(略)
Execution time: 0.018426 [seconds]
```

という天と地の差の違いがある。インタプリタを使う実行環境と，コンパイラによるネイティブアプリの速度さを体感してほしい。

課題 E

フォームから次元数を入力し，行列・ベクトル積を計算するよう matmul.php スクリプト (HTML と一体化しても良いし，HTML ファイルにフォームを書いても良い) を改変せよ。

第 5 章の学習チェックリスト

- □ PHP の基本文法をある程度理解し，簡単な計算を行うスクリプトを作成できる。
- □ HTML のフォームタグの機能を理解し，GET メソッドと POST メソッドを用いてフォームから入力されたデータを PHP スクリプトに渡すことができる。
- □ HTML フォームを持った PHP スクリプトを作成し，入力されたデータを自在に加工することができる。

[*2] Intel Core i7 820 + CentOS 5.5 x86_64 環境。