

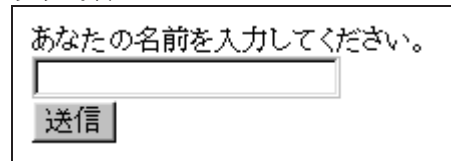
第2章 Form と CGI

2.1 Form

HTML のページの中に、文字列を入力するための場所やチェックボックスなどが埋め込まれていることがある。この入力のための領域がフォームと呼ばれる部分である。

フォームの例:

ファイル Aisatsu.html



```
<FORM ACTION="Aisatsu.cgi" METHOD="POST">
あなたの名前を入力してください。<BR>
姓<INPUT TYPE="text" SIZE="10" NAME="family">
名<INPUT TYPE="text" SIZE="10" NAME="given"><BR>
<INPUT TYPE="submit" VALUE="送信">
</FORM>
```

フォームは全体を `<FORM ... >~</FORM>` というタグで囲む。その中に `<INPUT ... >` などのタグを使用する。

- `<FORM ACTION="URL" METHOD="POST">~</FORM>`

`URL` は、このフォームのデータを受け取る CGI の URL である。

なお、`METHOD="POST"` ではなく、`METHOD="GET"` とすると、以前に紹介した URL の最後に ? をつけてパラメータを渡す方式で CGI にデータを渡すことになる。しかし GET では受け渡しできるデータの大きさに限界があるので、フォームでは POST を使うことが多い。

- `<INPUT TYPE="text" SIZE="n" NAME="nae">`

テキストボックスを表示する。テキストボックスは文字列を入力するための領域で、フォームの中で一番良く用いられる部品だと思われる。`n` は、このテキストボックスの幅、`nae` は、このテキストボックスを識別するための名前である。なお `TYPE="text"` を `TYPE="password"` に変えるとパスワードを入力するためのテキストボックス (入力した文字が伏せ字 (*) になる) を表示する。

- `<INPUT TYPE="checkbox" NAME="nae" VALUE="str" >`

チェックボックスを表示する。`str` はこのチェックボックスがチェックされていたときに、CGI に送る文字列であり、この `VALUE` 属性が省略されているときは、“on” という文字列を送る。また `CHECKED` という属性がついていると最初からチェックされている状態で表示する。

- `<INPUT TYPE="radio" NAME="nae" VALUE="str">`

ラジオボタンを表示する。ラジオボタンはチェックボックスに似ているが、`nae` が同じラジオボタ

ンはそのうち一つしか選択できない。*str* はこのラジオボタンが選択されていたときに、CGI に送る文字列である。CHECKED 属性がついていると最初からチェックされている状態で表示する。

- `<INPUT TYPE="hidden" NAME="naemae" VALUE="str">`

hidden は画面には表示されないが、名前と値は CGI に転送される。

- `<INPUT TYPE="submit" VALUE="str">`

送信ボタンを表示する。このボタンが押されると CGI にフォームのデータを転送する。*str* はこのボタンに表示する文字列である。

- `<INPUT TYPE="reset" VALUE="str">`

リセットボタンを表示する。このボタンが押されるとフォームに記入した内容をクリアする。*str* はこのボタンに表示する文字列である。

- `<TEXTAREA COLS="haba" ROWS="takasa" NAME="naemae">~</TEXTAREA>`

複数行の文字が入力できるテキストボックスを表示します。*haba* は幅、*takasa* は高さを指定する。~の部分の文字列が、このテキストボックスに最初に表示される。

2.2 CGI へのパラメータ渡し (POST 編)

POST で データを受け取る場合、CGI プログラム側はその入力を標準入力から受け取る。また、CONTENT_LENGTH という環境変数から、そのデータのバイト数を知ることができる。Java の場合、(QUERY_STRING の時と同様) シェルスクリプトが java コマンドにこの環境変数を渡してやることにする。Java のプログラム中では、System.getProperty("CONTENT_LENGTH") でこの値を知ることができる。

つまり Form の内容を answer という変数に読み込むためには、次のようにすれば良い。

```
int len = Integer.parseInt(System.getProperty("CONTENT_LENGTH"));
byte[] buf = new byte[len];
System.in.read(buf);
String answer = new String(buf, "ASCII");
```

この時の answer は次のような形の文字列になっている。

```
name1=value1&name2=value2& ... &namen=valuen
```

name₁, *name₂*, ... は INPUT タグや TEXTAREA タグに付けられていた NAME 属性で *value₁*, *value₂*, ... はそれぞれに対応する値 (テキストボックスの場合は入力された文字列、チェックボックスやラジオボタンなどの場合は、タグ中の VALUE 属性) である。

Java では StringTokenizer クラス¹を利用して必要な値を取り出すと良い。まず、& を StringTokenizer の区切り文字として、*name=value* のペアを取り出し、次に = を StringTokenizer の区切り文字として、NAME 属性と値を読む。

¹<http://guppy.eng.kagawa-u.ac.jp/2000/Enshu2/docs/ja/api/java/util/StringTokenizer.html>

なお、フォームに日本語を使っている場合、日本語は特別なエンコーディングをされて送られてくる。例えば“香川大学”は“%8D%81%90%EC%91%E5%8Aw”とエンコードされる (Shift JISの場合)。そこで、これをデコードする必要がある。

例題: おうむ返し

さきほどの Aisatsu.html のフォームを処理する CGI である。フォームに入力された内容を、そのままおうむ返しに表示する。とりあえずフォームに日本語は入力しないものとする。

ファイル Aisatsu.java

```
import java.io.*;
import java.util.*; // StringTokenizer 用に必要

class Aisatsu {

    /* ① */

    static public void main(String[] args) {
        try {
            String slen = System.getProperty("CONTENT_LENGTH");
            String family="", given="";
            if (slen!=null) {
                int len = Integer.parseInt(slen);
                byte[] buf = new byte[len];
                System.in.read(buf);
                String answer = new String(buf, "ASCII");
                StringTokenizer st = new StringTokenizer(answer, "&");
                while (st.hasMoreTokens()) {
                    StringTokenizer st1 = new StringTokenizer(st.nextToken(), "=");
                    String name = st1.nextToken();
                    if (name.equals("family")) {
                        family = st1.nextToken(); /* ② */
                    } else /* if (name.equals("given")) */ {
                        given = st1.nextToken(); /* ② */
                    }
                }
            }

            PrintWriter stdout =
                new PrintWriter(new OutputStreamWriter(System.out, "SJIS"));
            stdout.println("Content-type: text/html");
            stdout.println ();
            stdout.println("<HTML><HEAD></HEAD><BODY>");
            stdout.println("こんにちは、 "+family+" "+given+"さん!");
            stdout.println("</BODY></HTML>");
            stdout.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

フォームに日本語が入っている場合に対応するためにはデコードをする必要がある。内容は詳しく説明しないが次のようなメソッドの定義を /* ① */ の部分に挿入し、 /* ② */ の行を、

```
... = decodeURL(st1.nextToken());
```

に変えればデコードができる²

```
static int decodeURLAux(String str, byte[] buf, int offset) {
    // Formのデータを decode
    int len = str.length();
    if (len==0) {
        return offset;
    } else {
        char c = str.charAt(0);
        if (c=='+') {
            buf[offset] = (byte)32;
            return decodeURLAux(str.substring(1), buf, offset+1);
        } else if (c=='%') {
            String hex = str.substring(1, 3);
            buf[offset] = (byte)(Integer.parseInt(hex, 16));
            return decodeURLAux(str.substring(3), buf, offset+1);
        } else {
            buf[offset] = (byte)c;
            return decodeURLAux(str.substring(1), buf, offset+1);
        }
    }
}

static String decodeURL(String str) {
    try {
        byte[] buf = new byte[str.length()];
        int len = decodeURLAux(str, buf, 0);
        return new String(buf, 0, len, "JISAutoDetect");
    } catch (UnsupportedEncodingException e) { return "error!"; }
}
```

問 2.2.1 (見積り作成 CGI)

品名と単価の表と、その個数を入力してもらうためのフォーム(テキストボックス)を表示する HTML ファイルと、そのフォームから入力を受け取って、合計金額を表示する CGI を作成せよ。(さらに結果を見積書のようにテーブルとして整形せよ。)

必要な個数を入力してください。

品名	単価	個数
フロッピーディスク	50円	<input type="text"/>
CD-R	100円	<input type="text"/>
A4用紙 500枚	400円	<input type="text"/>

問 2.2.2 (1章の HiLite.cgi の拡張)

右のようなフォームから入力を受け取って、あるファイル中の指定された文字列を、指定された色で強調して表示する CGI を作成せよ。

赤色にする文字列	<input type="text"/>
緑色にする文字列	<input type="text"/>
青色にする文字列	<input type="text"/>

²本来、java.net.URLDecode.decode(String) というメソッドがこの処理をやってくれるはずだが、(今のところ)日本語コードの変換がうまくできないようだ。

例題: ゲストブック

Web ページを見た人に名前やメールアドレス、感想などを記入してもらい、HTML ファイルに保存する CGI である。フォームの HTML のソースはつぎのようになる。

ファイル GuestBook.html

```
<HTML><HEAD><TITLE>ゲストブック記帳</TITLE></HEAD>
<BODY>
<FORM ACTION="GuestBook.cgi" METHOD="POST">
ゲストブックに記帳をお願いします。<BR>
<TABLE>
<TR><TD>名前:</TD><TD><INPUT TYPE="text" SIZE="30" NAME="名前"></TD></TR>
<TR><TD>メールアドレス:</TD>
  <TD><INPUT TYPE="text" SIZE="30" NAME="メールアドレス"></TD></TR>
<TR><TD>ホームページ:</TD>
  <TD><INPUT TYPE="text" SIZE="30" NAME="ホームページ"></TD></TR>
<TR><TD>何かひとこと</TD>
  <TD><TEXTAREA NAME="ひとこと" ROWS="5" COLS="30"></TEXTAREA></TD>
</TABLE>
<INPUT TYPE="submit" VALUE="送信"><INPUT TYPE="reset" VALUE="やめ">
</FORM>
</BODY>
</HTML>
```

CGI プログラムでは、tmp というディレクトリにある Guests.html というファイルの最後の方にフォームに入力された内容を書き足していくことにする。一度、tmp.html という名前のファイルで変更した内容を作成し、あとで tmp.html のファイル名を Guests.html に変更する。

最初 Guests.html は次のような内容で作成し、誰でも書き込み可能にしておく。また tmp.html という一時ファイルを作成できるように、ディレクトリ tmp を書き込み可能にしておく。

ファイル Guests.html

```
<HTML><HEAD><TITLE>ゲストブック</TITLE></HEAD>
<BODY>
<H1 ALIGN="CENTER">ゲストブック</H1>
<HR>

<!-- OWARI -->
</BODY>
</HTML>
```

プログラムは長いのでいくつかに分けて提示する。

ファイル GuestBook.java (その1)

```
import java.io.*;
import java.util.*; // StringTokenizer 用に必要

class GuestBook {

    /* decodeURL の定義を挿入 */
```

最初の方は特にこれまでのプログラムと違う点はない。

ファイル GuestBook.java (その2)

```
static public void main(String[] args) {
    try {
        File f = new File("tmp/Guests.html");
        BufferedReader in = new BufferedReader
            (new InputStreamReader(new FileInputStream(f), "SJIS"));

        File tmp = new File("tmp/tmp.html");
        PrintWriter out = new PrintWriter
            (new OutputStreamWriter(new FileOutputStream(tmp), "SJIS"));
        while (true) {
            String line = in.readLine();
            if (line.trim().equals("<!-- OWARI -->")) break;
            out.println(line);
        }
        // 続く ...
    }
```

(その2)では、Guests.html, tmp.htmlの2つのファイルをオープンし、“<!-- OWARI -->”という行が現れるまでは、単にGuests.htmlからtmp.htmlへコピーをする。

ファイル GuestBook.java (その3)

```
// ...
String slen = System.getProperty("CONTENT_LENGTH");
if (slen!=null) {
    int len = Integer.parseInt(slen);
    byte[] buf = new byte[len];
    System.in.read(buf);
    String answer = new String(buf, "ASCII");
    StringTokenizer st = new StringTokenizer(answer, "&");
    out.println("<TABLE BORDER>");
    while (st.hasMoreTokens()) {
        out.print("<TR>");
        StringTokenizer st1 = new StringTokenizer(st.nextToken(), "=");
        String left = decodeURL(st1.nextToken());
        String right = st1.hasMoreTokens()?decodeURL(st1.nextToken()):"なし";
        out.print("<TD>"+left+"</TD>");
        out.print("<TD>"+right+"</TD>");
        out.print("</TR>");
    }
    out.println("</TABLE>");
    out.println("<HR>");
}
```

ここでフォームからのデータを解析し、テーブルを作成している。 $e_1?e_2:e_3$ は、 e_1 が真の時は e_2 の値、偽の時は、 e_3 の値を返す 3 項の Java の演算子 (C にもある) である。

ファイル GuestBook.java (その4)

```
out.println("<!-- OWARI -->");
while (true) {
    String line = in.readLine();
    if (line==null) break;
    out.println(line);
}
in.close();
out.close();
f.delete();
tmp.renameTo(f);
```

(その4)では CGIを起動したときに書込む部分の目印になるように、“<!-- OWARI -->”と出力し、Guests.htmlの残りの部分をtmp.htmlにコピーする。最後に両方をclose()して、Guests.htmlを削除し、tmp.htmlの名前をGuests.htmlに変更している。(この場合、Guests.htmlの持ち主がnobodyになってしまい、エディタから内容を変更できなくなるが、一度新しいファイルにコピーしてから編集し、そのファイルの名前をGuests.htmlに変更すれば良い。)

ファイル GuestBook.java (その 5)

```

PrintWriter stdout =
    new PrintWriter(new OutputStreamWriter(System.out, "SJIS"));
stdout.println("Content-type: text/html");
stdout.println ();
stdout.println("<HTML><HEAD></HEAD><BODY>");
stdout.println("御記帳有難うございました。<BR>");
stdout.print("ゲストブックは<A HREF=¥"tmp/Guests.html¥">こちら</A>です。");
stdout.println("</BODY></HTML>");
stdout.close();
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

お礼とできあがったゲストブックへのリンクをブラウザに表示して実行を終える。

問 2.2.3 (日記作成 CGI)

「日付」と「天気」と「題名」と「日記」の 4 つの入力ができる日記作成 CGI (*Diary.cgi*) を作成せよ。ゲストブックと逆に、新しい日記ほど前に付け加えられるようにせよ。

問 2.2.4 (家計簿)

以下の項目を入力してください

日付	<input type="text"/> 月 <input type="text"/> 日
食費	<input type="text"/> 円
衣料費	<input type="text"/> 円
娯楽費	<input type="text"/> 円
教育費	<input type="text"/> 円
雑費	<input type="text"/> 円
<input type="button" value="送信"/>	

右のようなフォームから入力を受け取って、簡単な家計簿を生成する CGI を作成せよ。その日だけの支出の計とそれまでの支出の累計の両方を計算するようにせよ。

例題: QUIZ

ここで紹介する CGI は次のようなテキストファイル

ファイル quiz.txt

```

日本で一番高い山は? エベレスト 富士山 飯野山 2
日本で一番広い湖は? 琵琶湖 府中湖 満濃池 1
シドニーで一番金メダルを取った国は? 日本 豪州 米国 3
工学部の所在地は? 幸町 花園町 林町 3

```

から次のような QUIZ を表示する HTML ページ


```

ようこそ QUIZへ!
では最初の問題です。

問: 日本で一番高い山は?

 エベレスト  富士山  飯野山
 

```

を作成する CGI である。簡単のため問題は 3 択に固定している

この CGI の特徴的なところは、フォームの表示されない隠し要素 `<INPUT TYPE="hidden" VALUE="... ">` を使って、同じ CGI プログラムに異なる表示をさせるところにある。“現在何番目の問題か?” (number) と “これまでの正解数” (score) が隠し要素の VALUE 属性として使用されている。

ファイル Quiz.java (その 1)

```

import java.io.*;
import java.util.*; // StringTokenizer 用に必要

class Quiz {

    /* decodeURL の定義はここ */

```

(その 1) の部分は特に変わったところはない。

ファイル Quiz.java (その 2)

```

static public void main(String[] args) {
    try {
        int number=0, score=0, answer=0;
        String message;
        File f = new File("Quiz.txt");
        BufferedReader in = new BufferedReader
            (new InputStreamReader(new FileInputStream(f), "SJIS"));
        String line="";
        int i;
        String lenS = System.getProperty("CONTENT_LENGTH");
        if (lenS==null) { // 最初の問か?
            message = "ようこそ QUIZ へ!<BR>では最初の問題です。<P>";
        }
    }
}

```

いくつかの局所変数を宣言し、QUIZ のデータが入っているファイル (Quiz.txt) を開く。そして lenS に環境変数 CONTENT_LENGTH の値を代入する。最初に CGI が実行されるときは、この lenS が null になっているはずなので、最初の問題を表示するようにする。

ファイル Quiz.java (その 3)

```

else { // 最初の間ではない場合
int len = Integer.parseInt(lenS);
byte[] buf = new byte[len];
System.in.read(buf);
String response = new String(buf, "ASCII");
StringTokenizer st = new StringTokenizer(response, "&");
while (st.hasMoreTokens()) {
StringTokenizer st1 = new StringTokenizer(st.nextToken(), "=");
String key = decodeURL(st1.nextToken());
int value = Integer.parseInt(decodeURL(st1.nextToken()));
if (key.equals("number")) {
number = value; // hidden からデータを読む
} else if (key.equals("score")) {
score = value; // hidden からデータを読む
} else if (key.equals("answer")) {
answer = value;
}
}
}

```

lenS が null でないときは、CGI がフォームから実行された（つまり最初の問題でない）ということなので、フォームから渡されたデータを解析する。number に “前の問題が何番目の問題か？”、score に “これまでの正解数” が入っている。answer は前の問題で解答者が選んだ答の番号である。

ファイル Quiz.java (その 4)

```

for (i=0; i<number; i++) {
line = in.readLine(); // number-1 行分、読み飛ばす
}
line = line.trim(); // trim() は前後の空白を除去する
int a = Integer.parseInt(line.substring(line.length()-1));
if (a==answer) { // a は最後の文字
message = "正解です。<BR>";
score++;
} else {
message = "残念でした。<BR>";
}
} // else (最初の間ではない場合) の終

```

number-1 行分読み飛ばして、number 行目の最後の文字を解答と比べる。その結果によってメッセージと score 変数の値を変える。

ファイル Quiz.java (その 5)

```

PrintWriter stdout =
new PrintWriter(new OutputStreamWriter(System.out, "SJIS"));
stdout.println("Content-type: text/html");
stdout.println ();
stdout.println("<HTML><HEAD></HEAD><BODY>");
stdout.println(message);
line = in.readLine();
in.close();
if (line==null || line.trim().equals("")) { // 終
stdout.println("<BR>これで QUIZ は終わりです。<BR>");
stdout.println("正解数は、"+score+"問でした。");
}
}

```

次の行を読み込む。(line = in.readLine()) line が null か空文字列なら、そこで Quiz.txt は

終なのでクイズを終了する。

ファイル Quiz.java (その6)

```
else { // 終ではない
    StringTokenizer st1 = new StringTokenizer(line);
    stdout.println("次の問: "+st1.nextToken()+"<BR>");
    stdout.println("<FORM METHOD=%"POST%">");
    stdout.println("<INPUT TYPE=%"hidden%" NAME=%"number%"");
    stdout.println(" VALUE=%"+(number+1)+"%">"); // hiddenにデータを保存
    stdout.println("<INPUT TYPE=%"hidden%" NAME=%"score%"");
    stdout.println(" VALUE=%"+score+"%">"); // hiddenにデータを保存
```

終でなければ、lineから次の問の情報を読み取って、フォームとして出力する。number+1とscoreの値を新しい隠し要素のVALUE属性として出力する。これらの値は次にCGIプログラムが起動されるときに使用される。

つまり、QUIZの各問は別々に実行され、表示されるので、これらの値をサーバ側で保存しておくことはできない。それで、隠し要素(hidden)の属性としてWebブラウザ側に渡してしまうのである。

そして最後に各選択肢のラジオボタンと送信ボタン、リセットボタンを出力する。

ファイル Quiz.java (その7)

```
for (i=0; i<3; i++) {
    stdout.print("<INPUT TYPE=%"radio%" NAME=%"answer%"");
    stdout.print(" VALUE=%"+(i+1)+"%">");
    stdout.print(" "+st1.nextToken());
}
stdout.println("<BR>");
stdout.println("<INPUT TYPE=%"submit%" VALUE=%"送信%">");
stdout.println("<INPUT TYPE=%"reset%" VALUE=%"やめ%">");
stdout.println("</FORM>");
} // else(終ではない)の終
stdout.println("</BODY></HTML>");
stdout.close();
} catch (Exception e) {
    e.printStackTrace();
}
}
```

問 2.2.5 Quizを3択だけではなくて任意の数の選択肢から選べるように拡張せよ。

問 2.2.6 (アンケート)

Quizを、正解数だけではなくて、どの問に対してどの選択肢を選んだかまでわかるように記録し、その結果をファイルに書き出すように変更せよ。

参考: POSTを利用するCGIプログラムをデバッグするときには、フォームから送られてくるデータを保存しておくると便利である。それには次のようなCGIを利用すれば良い。

ファイル Debug.java

```
import java.io.*;

class Debug {
    static public void main(String[] args) {
        try {
            PrintWriter stdout =
                new PrintWriter(new OutputStreamWriter(System.out, "SJIS"));
            stdout.println("Content-type: text/html");
            stdout.println ();
            stdout.println("<HTML><HEAD></HEAD><BODY>");
            String cl = System.getProperty("CONTENT_LENGTH");
            if (cl==null) {
                String qs = System.getProperty("QUERY_STRING");
                stdout.println("<TT>QUERY_STRING</TT>は "+qs+"です。");
            } else {
                int len = Integer.parseInt(cl);
                byte[] buf = new byte[len];
                System.in.read(buf);
                String answer = new String(buf, "ASCII");
                File f = new File("tmp/Debug.out");
                PrintWriter out = new PrintWriter(new FileWriter(f));
                out.print(answer);
                out.close();
                stdout.println("<TT>CONTENT_LENGTH</TT>は "+len+"です。");
                stdout.println("<A HREF=¥"tmp/Debug.out¥">tmp/Debug.out</A>にデバッグ情報
を書きこみました。");
            }
            stdout.println("</BODY></HTML>");
            stdout.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

この CGI はフォームから送られてくる情報を Debug.out というファイルに書き出し、同時に CONTENT_LENGTH の値をブラウザに表示する。

キーワード:

フォーム、POST、CONTENT_LENGTH、hidden