第1章　CGIプログラミング

1.1　CGIとは

CGI（**C**ommon **G**ateway **I**nterface）は、Webサーバ上でプログラムを実行し、動的にHTML形式などのデータなどを作成してWebブラウザに渡すための仕組みである。

Javaのアプレットや JavaScriptはクライアント（Webブラウザが実行されているコンピュータ）側でプログラムが実行されるのに対し、CGIはサーバ（HTMLの文書が置かれているコンピュータ）側でプログラムが実行されるという違いがある。例えばアクセスカウンタはクライアント上のプログラムだけでは実現できないので CGIが必要になる。

CGIを記述する言語は何でもかまわないわけだが、Perlや Cを使うことが多いようである。PHP（`http://www.php.net/`）のように HTML中にスクリプトを埋め込むタイプもある。Perlが CGIに広く用いられている理由としては、

- インタプリタ方式で実行されるのでコンパイルという作業が不要であること
- インタプリタ方式の割に高速であること
- 正規表現や連想配列などが容易に扱えること
- hereドキュメントという形で文字列をプログラム中に埋め込むことが簡単であること

などが挙げられる。しかし Perlは習得が容易であるとは言えず、デバッグがちょっとした職人芸になる。

C言語は高速に実行できるが、CGIのプログラムに特に必要となる文字列操作が苦手である。

この演習では CGIの作成に Java言語を用いる。Javaは必ずしも CGIのプログラミングに向いているとは言いきれないところがあるが、アプレットの作成にも用いるため馴染みが深いし、少なくとも文字列処理は比較的 (?) 得意である。

有用なリンク

HTMLのまとめ

- 初めての ホームページ講座 （`http://www.hajimeteno.ne.jp/`）
- UmeJam HTML Help （`http://www.abacuss.com/`）

Java

- Java 2ドキュメント（Sun）
  （`http://java.sun.com/products/jdk/1.2/ja/docs/ja/`）
- Java 2ドキュメント（香川大学工学部内のコピー）
  （`http://guppy.eng.kagawa-u.ac.jp/2000/Enshu2/docs/ja/`）
- Java Tips（`http://www.asahi-net.or.jp/~dp8t-asm/java/tips/`）

1.2 CGI の作成と設置

CGI は、単純に言ってしまえば、HTML のデータ[1]を標準出力に出力するプログラムである。（入力を受け取る方法は後述する。）拡張子は通常（Web サーバの設定にもよるが）.cgi である。

ただし、単に HTML のデータを出力するだけでは、それを Web ブラウザに渡したときに、ブラウザが HTML のデータであるということが拡張子から判別できない。そこで、必ずデータの種類を表わす文字列（“Content-type: text/html”）をデータの先頭に付け加える。

ファイル MyDate.java

```
import java.io.*;
import java.util.*;

class MyDate {
  static public void main(String[] args) {
    Date d = new Date();
    System.out.println("Content-type: text/html");
    System.out.println ();  // 空行を出力
    System.out.println("<HTML><HEAD></HEAD><BODY>");
    System.out.println(d.toString());
    System.out.println("</BODY></HTML>");
  }
}
```

このプログラムは現在の時刻を表示する。このなかの

```
    System.out.println("Content-type: text/html");
    System.out.println ();  // 空行を出力
```

が、以下に続くデータが HTML のデータであるということをブラウザに伝える役目を持つ。（この “Content-type: text/html” の次には必ず空行が必要である。）

このソースは次のコマンドでコンパイルする。

```
javac -encoding SJIS MyDate.java
```

“SJIS” のところは、コンパイルする Java のソースファイルの日本語コードによって、SJIS, JIS, EUCJIS のいずれかを選択する。自分の使用する日本語コードは何か一つに固定しておくと良い。（以下では Shift JIS を使用すると仮定する。）

これで、MyDate.class というファイルができる。周知のように、.class ファイルの中身は Java の中間言語のコードであり、これを実行するには、java という中間コードの実行系が必要である。

```
java MyDate
```

.cgi ファイルは直接実行可能である必要があるので、Perl や C の場合のように .class ファイルを単に名前を変えるだけ、というわけにはいかない。ここでは .cgi ファイルは java を起動するシェルスクリプトを用いる。

ファイル MyDate.cgi

```
#!/bin/csh

exec /usr/local/jdk/bin/java MyDate
```

[1] もちろん JPEG や PNG などを出力する CGI があっても構わないが、この演習では扱わない。

他の名前のクラスを実行するときは一番最後の MyDate だけを変えれば良い。.cgi ファイルは、**実行可能フラグ**を立てておく。CGI は Web サーバにより実行されるので**自分以外のユーザでも実行できる**ようにしておく必要がある。

```
chmod a+x MyDate.cgi
```

参考:

演習ではこのようなシェルスクリプトを簡単に生成し、モードを変更するための Makefile を用意する。この Makefile があれば、make MyDate.cgi というコマンドで .cgi ファイルを作成できる。ちなみにこれは自分と拡張子を除いた部分が同じ名前の.class ファイルを実行するシェルスクリプトである。

ファイル Makefile

```
.SUFFIXES:      .java .class .cgi

.java.class:
        javac -encoding SJIS $<

.java.cgi:
        cat template.txt > $@
        echo "exec /usr/local/jdk/bin/java $*" >> $@
        chmod a+x $@
```

template.txt には（今のところ）

```
#!/bin/csh
```

の一行だけを書いておく。

これで CGI は完成である。用意したファイル（MyDate.class とMyDate.cgi）を Web サーバから見えるところ（設定により異なるが、~/public_html/cgi-bin/など）に置き、Web ブラウザからアクセスする。（~/public_html/cgi-bin/に置いた場合は、通常 URL は、
http://*XXX.YYY.ZZZ.QQQ*/~*login*/cgi-bin/MyDate.cgi になる。ただし、*XXX.YYY.ZZZ.QQQ* は Web サーバを実行しているコンピュータの名前または IP アドレス、*login* は自分のログイン名になる。）

問 1.2.1 上の *CGI*プログラムで時刻によって、ブラウザに表示されるときの色が変わるようにせよ。例えば、*18*〜 *6* 時が黒、*6*〜 *12* 時が青、*12*〜 *18* 時が赤など。

ヒント:

- *Java: java.util.Date* クラスの *getHours* メソッド
  (http://guppy.eng.kagawa-u.ac.jp/2000/Enshu2/docs/ja/api/java/util/Date.html)
- *HTML: <FONT COLOR="RED"> ... </FONT>*など

問 1.2.2 時刻によって、あるいは乱数によって、ブラウザに表示されるページの背景画像が表示されるたびに変わるようにせよ。

ヒント:

- *Java: java.util.Random* クラス
  (http://guppy.eng.kagawa-u.ac.jp/2000/Enshu2/docs/ja/api/java/util/Random.html)

- *HTML: <BODY BACKGROUND=" ... "> ... </BODY>*など
- 素材*: http://home.interlink.or.jp/~hrak/yakata/index.html*（素材の館）
  http://wakusei.cplaza.ne.jp/ushikai/（牛飼いとアイコンの部屋）

1.3 アクセスカウンタ

アクセスカウンタはもっとも代表的な CGI で、Web ページに対するアクセスの回数を記録し、表示するものである。アクセスの回数はサーバ上のファイルに記録しておく。

ファイル Counter.java

```
class Counter {
  static public void main(String[] args) {
    try {
      File f = new File("counter.txt"); // 本当はフルパスで指定する方が安全
      BufferedReader in = new BufferedReader(new FileReader(f)); // 入力用 open
      String str = in.readLine();
      int i = Integer.parseInt(str);
      in.close(); // 出力用に openする前に 必ず closeしておく。
      PrintWriter out = new PrintWriter(new FileWriter(f)); // 出力用に open
      out.println(++i);
      out.close(); // closeを忘れないこと

      // 文字コードを変換するために System.outの代わりにもちいる
      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("あなたは "+i+"番目の来訪者です。");
      stdout.println("</BODY></HTML>");
      stdout.close();  // closeを忘れない
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

この例では counter.txt というファイルにアクセス回数を記録している。このファイルは、Counter.cgi と同じディレクトリに置いておく。counter.txt の中身は 1 行のみの数字だけのファイルであり、最初は 0 を書いておく。CGI スクリプトがこの counter.txt の内容を書き換えることができるように、counter.txt は**誰でも読み書きが出来るようになっていなくてはならない。**

```
chmod a+w counter.txt
```

このコマンドを実行すると、counter.txt が誰でも（CGI プログラムも）読み書きができるようになる。

プログラム中の

```
File f = new File("counter.txt"); // 本当はフルパスで指定する方が安全
BufferedReader in = new BufferedReader(new FileReader(f));
...
in.close();
```

は、ファイルから入力するときの常套句で、... の部分では、inというオブジェクトに対して、System.inからの入力と同じようにファイルからの入力が可能になる。同様に、

```
PrintWriter out = new PrintWriter(new FileWriter(f));
...
out.close();
```

は、ファイルへ出力するときの常套句である。outというオブジェクトに対して、System.outに対するのと同じメソッドが使用できる。ここまでの部分で、ファイルから数字を読み込み、一つ増やした数字をファイルに書き込んでいる。

また、以下の部分

```
// 文字コードを変換するために System.outの代わりにもちいる
PrintWriter stdout =
    new PrintWriter(new OutputStreamWriter(System.out, "SJIS"));
```

は、日本語の文字コードを変換するためのコードである。Javaのプログラムの内部では日本語はUnicodeというコード系で表わされている。これを出力するときに、Shift JISに変換するためのフィルタをかけるようにする。（CGIの場合, 文字コードの扱いが通常と比べてなぜかシビアになるようである。）これに続く部分では System.outの代わりに、このstdoutを用いている。このstdoutについても最後にclose()するのを忘れないようにする。

最後に try～catchで全体を囲んでいる。この例では実行時の例外（例えば、ファイルが見つからなかった・ファイルに書き込みが許可されていなかったなどに起因するエラー）に対しては何も処置を行なわない。もちろん、実際には例外が起きた場所と種類によってきめ細かく処置をする方が良い。

ファイル Counter.cgiは、MyDate.cgi中の “MyDate” を “Counter” に書き換えればよい。もちろんchmod a+x Counter.cgiしておく。

問 1.3.1 カウンタの値が特別な値（例えば *10*の倍数など）になったときは、メッセージを変えたり、色を変えたりするように改造せよ。

1.png, 2.pngなどの名前で数字画像ファイルを用意しておいて、このアクセスカウンタや時刻表示CGIで 1, 2, と表示する代わりに<IMG SRC="1.png">, <IMG SRC="2.png">などにしておくと、数字を画像で表示するアクセスカウンタ・時刻表示 CGIができる。

問 1.3.2 数字を画像として表示するアクセスカウンタ・時刻表示 *CGI*を作成せよ。

参考: 数字画像データ

- *Digit Mania*（*http://www.digitmania.holowww.com*）
- *Counter Art*（*http://www.counterart.com/*）

参考: 画像を生成する *CGI*

- *gd*（*http://www.boutell.com/gd/*）

1.4 CGIへのパラメータ渡し（GET編）

これまで紹介した CGIはブラウザ側から CGIプログラムにデータを渡すことはなかったが、ブラウザから CGIにパラメータを渡して CGIの振舞いを変えることも可能である。

例題: キーワードのハイライト

キーワードをパラメータとして受け取り、特定のファイルを読み込んで、キーワードの部分を色を変えて表示する CGI（HiLite.cgi）を作成する。

CGI にパラメータを渡す方法には GET と POST の 2 種類がある。ここでは GET について説明する。

GET で CGI にパラメータを渡すには URL のあとに “?” に続けて文字列を書けば良い。この文字列の部分がパラメータとして CGI に渡される。例えば

http://*XXX.YYY.ZZZ.QQQ*/~*login*/cgi-bin/HiLite.cgi?print

の、print の部分がパラメータになる。

CGI プログラム中では、このパラメータは QUERY_STRING という環境変数（OS からプログラムに渡される変数）から受け取ることになる。ただし、C 言語や Perl の場合には環境変数にアクセスすることは簡単であるが、Java では環境変数をアクセスする方法が用意されていない。このため、java コマンドを起動するシェルスクリプトで環境変数を読んで、java コマンドに渡してやることにする。（下のスクリプトでは、QUERY_STRING 以外に今後必要となる環境変数をついでに渡している）

ファイル HiLite.cgi

```
#!/bin/csh

set d0=""
if ($?REQUEST_METHOD) then
   set d0="$d0 -DREQUEST_METHOD=$REQUEST_METHOD"
endif
if ($?CONTENT_LENGTH) then
   set d0="$d0 -DCONTENT_LENGTH=$CONTENT_LENGTH"
endif
if ($?QUERY_STRING) then
   set d0="$d0 -DQUERY_STRING=$QUERY_STRING"
endif

exec /usr/local/jdk/bin/java HiLite
```

（実際に演習で使用するスクリプトでは、デバッグのためにもう少し拡張して、“./HiLite -get *QUERY_STRING*” または “./HiLite -post *CONTENT_LENGTH*” のように起動して、環境変数を渡せることができるようになっている。）

参考:

シェルスクリプトを自動生成するための Makefile も次のように変更する。

```
.SUFFIXES:      .java .class .cgi

.java.class:
        javac -encoding SJIS $<

.java.cgi:
        cat template.txt > $@
        echo "exec /usr/local/jdk/bin/java ¥$$d0 $*" >> $@
        chmod o+x $@
```

また、template.txt は HiLite.cgi の最後の一行を除いたものになる。

このように java コマンドに -D オプションを用いて渡されたパラメータは Java のプログラム中では、System.getProperty() というメソッドで得ることができる。

ファイル HiLite.java

```
import java.io.*;
import java.util.*;

class HiLite {

  // substrReplaceの定義はここに入る

  static public void main(String[] args) {
    try {
      File f = new File("HiLite.java"); // 本当はフルパスで指定する方が安全
      String word = System.getProperty("QUERY_STRING");
      InputStreamReader fr = new InputStreamReader
                                  (new FileInputStream(f), "SJIS");
      BufferedReader in = new BufferedReader(fr);
      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("<PRE>");
      while(true) {
        String line = in.readLine();
        if (line==null) break;
        line = substrReplace(line, "<", "&lt;");
        line = substrReplace(line, ">", "&gt;");
        if (word!=null && word.length()!=0) {
            line = substrReplace(line, word,
                                 "<FONT COLOR=¥"RED¥">"+word+"</FONT>");
        }
        stdout.println(line);
      }
      stdout.println("</PRE>");
      stdout.println("</BODY></HTML>");
      stdout.close();
    } catch(Exception e) {
      e.printStackTrace();
    }
  }
}
```

結局、

```
String word = System.getProperty("QUERY_STRING");
```

の部分で、URLの “?” 以降の部分の文字列が変数 word に入ることになる。

```
line = substrReplace(line, word,
                     "<FONT COLOR=¥"RED¥">"+word+"</FONT>");
```

で、このキーワードを赤色にするように置換している。ここで、substrReplaceは文字列中の部分文字列を別の文字列に置換するメソッドである[2]。

```
static String substrReplace(String str, String str1, String str2) {
  // str中に含まれる str1を str2に置換する。
```

[2]この程度のメソッドがなぜjava.lang.Stringクラスに入っていないのかは不思議だ。

```
    // 例: substrReplace("conversation", "vers", "serv") => "conservation"
    int i = str.indexOf(str1);
    if (i==-1) {
      return str;
    } else {
      return str.substring(0, i)+str2+
        substrReplace(str.substring(i+str1.length()), str1, str2);
    }
  }
```

ところで、表示するテキストの中に “<” や “>” が入っていると、HTML のタグと解釈されてしまって、表示が乱れる。次の部分

```
  line = substrReplace(line, "<", "&lt;");
  line = substrReplace(line, ">", "&gt;");
```

は、これらを <, > にそれぞれ置き換えている。

結局、このプログラムは HiLite.java のソース自身（これはたまたまで、別のファイルにしてももちろん良い）の中のキーワードを赤色で表示することになる。

http://*XXX.YYY.ZZZ.QQQ*/~*login*/cgi-bin/HiLite.cgi?print

だと、print という部分文字列が赤色で表示されることになる。

例題: テキストの表から HTML の表へ

```
1 2 3
4 5 6
```

のような plain text の表を

```
<TABLE BORDER>
<TR><TD>1</TD><TD>2</TD><TD>3</TD></TR>
<TR><TD>4</TD><TD>5</TD><TD>6</TD></TR>
</TABLE>
```

のように HTML に変換する CGI プログラムである。plain text の表が入っているファイル名を CGI の引数として用いる。

ファイル Tabular.java

```
import java.io.*;
import java.util.*;   // StringTokenizer

class Tabular {
  static public void main(String[] args) {

    try {
      File f = new File(System.getProperty("QUERY_STRING"));
      BufferedReader in = new BufferedReader(new FileReader(f));
      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("<TABLE BORDER>");

      while(true) {
        String line = in.readLine();
        if (line==null) break;
        stdout.print("<TR>");
        StringTokenizer st = new StringTokenizer(line);
        while(st.hasMoreTokens()) {
          stdout.print("<TD>"+st.nextToken()+"</TD>");
        }
        stdout.println("</TR>");
      }
      stdout.println("</TABLE>");
      stdout.println("</BODY></HTML>");
      stdout.close();
    } catch(Exception e) {
      e.printStackTrace();
    }
  }
}
```

http://*XXX.YYY.ZZZ.QQQ*/~*login*/cgi-bin/Tabular.cgi?table.txt

のように使用する。

このプログラム中の、StringTokenizer クラスについては、

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

を参照して欲しい。

注意!

この CGIプログラムを実際に Web サーバに置くことは危険である。

一般に CGI の引数にファイル名を渡す場合には注意が必要である。悪意のある人が ?/etc/passwd のような引数を渡すと簡単にパスワードファイルを盗んだりすることが可能になる。CGI の引数にファイル名を許すときは先頭に “/” がないか（つまり絶対パスでないか）、.. のような文字が入っていないか、をなどをチェックする必要がある。同様に引数として渡された文字列をコマンドとして実行するような CGI も注意が必要である。

自作の CGI を設置する場合はセキュリティに関して細心の注意を払って欲しい。

問 1.4.1 上の例題に、上の注意でファイル名のチェックを追加せよ。また、パラメータが渡されなかった場合（System.getProperty("QUERY_STRING")が *null* になる）や、ファイルが存在しなかった場合などのチェックをきちんとするようにせよ。

問 1.4.2 （カレンダー）
例えば、*Calerdar.cgi?200010* のような形でパラメータを渡されると、*2000* 年 *10* 月のカレンダーを作成するような *CGI* を作成せよ。

ヒント： *y* 年 *m* 月 *d* 日の曜日を求めるのに次のような *Zellar* の公式を用いよ

```
static int Zellar(int y, int m, int d) {
  if (m<3) {   // 1月、2月は前年の 13月、14月として計算する。
    y--; m+=12;
  }
  return (y + y/4 - y/100 + y/400 + (13*m+8)/5 + d) % 7;
  // 0が日曜、1が月曜、... 6が土曜
}
```

問 1.4.3 （スライドショー）
ディレクトリ中に *1.png*, *2.png*, ... のような名前の画像ファイルを用意しておくと、この画像ファイルを順に表示するような *CGI*（*SlideShow.cgi*）を作成せよ。

ヒント： QUERY_STRINGがない（*System.getProperty*が *null* を返す）ときは *1.png* を表示する。QUERY_STRINGが *n* の時は、次のような *HTML* を生成する。

```
<HTML><HEAD><TITLE>スライド（n）</TITLE></HEAD><BODY>
<DIV ALIGN="CENTER">
<IMG SRC="n.png"><HR>
<A HREF="SlideShow.cgi?n-1">前</A>
<A HREF="SlideShow.cgi?n+1">次</A>
</DIV>
</BODY></HTML>
```

問 1.4.4 （ディレクトリ・リスティング）
あるディレクトリ（これは固定で良い）のインデックスを作成する *CGI*（*DirIndex.cgi*）を作成せよ。ただし、QUERY_STRINGで渡された数の日数より変更された日付が新しいファイルには *"NEW!"* などのマークをつけるようにせよ。

例えば、*old.txt* という *3* 日前に変更されたファイルと *new.txt* という *1* 日前に変更されたファイルがあるときは *DirIndex.cgi?2* は次のような *HTML* を出力する。

```
<HTML><HEAD><TITLE>ディレクトリ</TITLE></HEAD><BODY>
<UL>
<LI><A HREF="nex.txt">new.txt</A> NEW!
<LI><A HREF="old.txt">old.txt</A>
</UL>
</BODY></HTML>
```

ヒント：*java.io.File* クラス（*http://guppy.eng.kagawa-u.ac.jp/2000/Enshu2/docs/ja/api/java/io/File.html*）のメソッド、特に *list()* と *lastModified()* と *java.util.Date* クラスを用いる。

キーワード：

CGI, Perl, chmod, GET, 環境変数, QUERY_STRING