第3章　Javaアプレットと CGI

Java アプレットから CGI にデータを送ることも可能である。もちろん Java アプレットからサーバにデータを送る方法はいくらでも（例えば ftp サーバと接続する、IRC サーバと接続するなど）考えられるので、CGI 以外の方法が取れる場合は CGI との通信にこだわる必要は必ずしもない。

しかし、一般のプロバイダに Web ページを設置するとき、勝手にサーバを起動するようなことはできないのが普通なので、CGI と通信するのも手軽な方法である。

例えば、Java アプレットでゲームを作成し、ハイスコアを CGI でサーバに記録するような用途が考えられる。

3.1　GET による方法

アプレットから GET で送る方法は簡単である。URL クラス[1] のコンストラクタに URL を表わす文字列を渡すときに “?” のあとにパラメータをつけた形を用いれば良い。URL クラスを用いるには、最初にimport java.net.*;が必要である。

```
URL url = new URL("http://XXX.YYY.ZZZ.QQQ/~login/cgi-bin/HiLite.cgi?print");
getAppletContext().showDocument(url);
```

URL のコンストラクタは何種類か用意されているが、new URL(絶対パス) という形と、new URL(getCodeBase(), 相対パス) という形を良く使うと思われる。後者の形では、アプレットの .class ファイルのおいてあるディレクトリからの相対パスを用いることができる。

```
URL url = new URL(getCodeBase(), "HiLite.cgi?print");
```

AppletContext().showDocument(*url*)は、アプレットから Web ブラウザに指令を送り、画面を新しい URL（*url*）に切り替える。

例題: 色選択アプレット
この例題ではアプレットで実際の色を見ながら色を選択し、次に表示するページの背景の色を変えることができるようにしている。

まず、アプレット側のソースから紹介する。

[1] http://guppy.eng.kagawa-u.ac.jp/2000/Enshu2/docs/ja/api/java/net/URL.html

ファイル ColorSelect.java（その 1）

```
import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import java.net.*;      // URL用に必要

public class ColorSelect extends Applet implements MouseListener {
  public void init() {
    addMouseListener(this);
  }

  public void mouseEntered(MouseEvent me) {}
  public void mouseExited(MouseEvent me) {}
  public void mousePressed(MouseEvent me) {}
  public void mouseReleased(MouseEvent me) {}
```

（その 1）の部分はマウスイベントを利用するアプレットとして、ごく普通である。ただ、あとで URL クラスを利用するため、import java.net.*;がある。

ファイル ColorSelect.java（その 2）

```
  public void paint(Graphics g) {
    int r, y;
    for (r=0; r<16; r++) {
      for (g=0; g<16; g++) {
        Color c = new Color(r*16, g*16, 255);
        g.setColor(c);
        g.fillRect(r*16, g*16, 16, 16);
      }
    }
  }
  public void mouseClicked(MouseEvent me) {
     int x = me.getX(); int y = me.getY();
     if (0<=x && x<=255 && 0<=y && y<=255) {
       String message = Integer.toHexString(x*0x10000+y*0x100+0xff);
       if (x==0) { message = "00"+message; }
       else if (x<0x10) { message = "0"+message; }
       try {
         URL url = new URL(getCodeBase(), "ChangeBG.cgi?"+message);
         getAppletContext().showDocument(url);
       } catch (Exception e) {
          e.printStackTrace();
       }
     }
  }
}
```

paint メソッドは、光の三原色のうち、赤と緑の成分を変化させてグラデーションを描いている。mouseClicked メソッドではマウスが押された地点の色を計算し、ChangeBG.cgi という CGI に送っている。

なお、このアプレットを表示するための HTML ファイルとして、次のようなものを用意する。ファイル ColorSelect.html

```
<HTML><HEAD></HEAD>
<BODY>
マウスでクリックして色を選択してください。<BR>
<APPLET CODE="ColorSelect.class" WIDTH="256" HEIGHT="256">
</APPLET>
</BODY>
</HTML>
```

CGI 側（ChangeBG.java）は HiLite.java によく似ていて、特に新しいところはない。データを送ってくるのがアプレットの場合も、特に変更すべきところはない。

ファイル

```
import java.io.*;

class ChangeBG {
  substrReplace の定義は省略
  static public void main(String[] args) {
    try {
      File f = new File("ColorSelect.java");
      String color = 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>");
      stdout.println("<BODY BGCOLOR=¥"#" + color + "¥">");
      stdout.println("<PRE>");
      while(true) {
        String line = in.readLine();
        if (line==null) break;
        line = substrReplace(line, "<", "&lt;");
        line = substrReplace(line, ">", "&gt;");
        stdout.println(line);
      }
      stdout.println("</PRE>");
      stdout.println("</BODY></HTML>");
      stdout.close();
    } catch(Exception e) {
      e.printStackTrace();
    }
  }
}
```

ColorSelect.java のソースを表示するようにしているが、もちろん他のファイルでも良い。color という変数にアプレットから送られてきた文字列が代入される。この color を BODY タグの BGCOLOR 属性に用いて、ページの背景色を変えている。

注: アプレットのプログラムを書き換えたとき、ブラウザに新しいプログラムを読み込ませるためには、単に再読み込み（Reload）ボタンを押すだけではダメで、**Shift キーを押しながら**再読み込み（Reload）ボタンを押す必要がある。

問 3.1.1 背景色だけでなく、前景色（文字の色）も選択して変更できるようにせよ。また、色の選択の方法も、より広い範囲の色が選べるように工夫してみよ。

問 3.1.2 前景色（文字の色）を *2* つ選ぶと、その *2* つの色の間でグラデーションするようにせよ。

3.2 POSTによる方法

GETは CGIへ送るデータがユーザにも見えるので、例えばゲームのハイスコアを記録するような用途では “ずる” がしやすいという欠点がある。また、大量のデータを送るような用途にも向かない。

GETの場合ほどは簡単ではないが、 POSTによって Javaアプレットから CGIにデータを送ることも可能である。

POSTで CGIにデータを送るのは次のような手順で行なう。まず次のように CGIの URL（*url*）を指定して、URLオブジェクトを作成する。

```
URL url = new URL(url);
// または、URL url = new URL(url1, path);
```

続いて、この URLへの接続を確立し、データを送る準備をする。

```
URLConnection  urlCon = url.openConnection();
urlCon.setDoOutput(true);
urlCon.setDoInput(true);
urlCon.setAllowUserInteraction(false);
```

実際にデータを送って、最後に接続を閉じる。

```
PrintWriter writer = new PrintWriter(urlCon.getOutputStream());
  // この間に、writer.print(string) という形で writerへデータを送る。
writer.close();
```

続けて、CGIの出力を受け取る。

```
BufferedReader mReader = new BufferedReader(
                           new InputStreamReader(
                             urlCon.getInputStream(), "SJIS"));
  // この間、 mReader.readLine() という形で CGIからデータを受け取る。
mReader.close();
```

GETの場合のように、getAppletContext().showDocument(url)という形で、ブラウザに CGIの出力を表示させることはできない。その場合には工夫が必要である。

例題: タッチタイピングゲーム

タッチタイピング練習アプレットである。記録を CGIに送ってサーバ側で記録できるようにする。まずアプレット側のプログラムを紹介する。

ファイル TypingTest.java（その 1）

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;      // URL用に必要
import java.io.*;

public class TypingTest extends Applet implements Runnable, ActionListener {
  Label l0, l1;
  TextField tf;
  Thread thread;

  String[] data = {"rush", "push", "ball", "text", "aunt",
                   "help", "null", "dash", "zinc", "bomb"};
  String kotae="", message="";
  int score=0, count=100;
  boolean next = false;

  public void init() {
    l0 = new Label("        "); add(l0);
    tf = new TextField("", 5);    add(tf);
    l1 = new Label("                                    "); add(l1);
    tf.addActionListener(this);
    tf.requestFocus();
    thread = new Thread(this);
    thread.start();
  }
```

（その 1）の部分は、import 文とインスタンス変数の定義、init メソッドの定義である。import 文は通常のアプレットに必要なものに加えて、java.net.*と java.io.*を使用する。TypingTest クラスは、スレッドとボタンを用いるので、Runnable と ActionListener の 2 つのインターフェースをimplements する。

init メソッド中では、問題とメッセージをそれぞれ表示するための 2 つのラベル l0, l1 と、答を入力するためのテキストフィールド tf を用意している。

ファイル TypingTest.java（その 2）

```
  public void actionPerformed(ActionEvent ae) {
     String kotae1  = tf.getText();
     if (kotae1.equals(kotae)) {
       message+=kotae+"="+count+"&";
       score += count;
       count = 100;
       next = true;
     }
  }
```

actionPerformedはテキストフィールドで改行キーが押されたときの処理である。答と正解を比べて、正しければスコアをキー入力にかかった時間に応じて増分する。

ファイル TypingTest.java（その 3）

```
  public void run() {
    Thread thisThread = Thread.currentThread();

    int i;
    for (i=0; i<data.length; i++) {
      kotae = data[i];
      l0.setText(kotae);
      next = false;
      while(!next) {
        try {
          Thread.sleep(25);  // 25ミリ秒お休み
        } catch (InterruptedException e) {}
        if (count>0) count--;
        l1.setText("Score: "+count+".");
      }
      tf.setText("");
    }
```

run メソッドの前半部分（繰り返しの部分）である。data 配列中の問題を次々と表示していく。25 ミリ秒毎に点数（count）を減らしていく。

ファイル TypingTest.java（その 4）

```
    l1.setText("Total score is "+score+"!");
    message+="total="+score;
    try {
      URL url = new URL(getCodeBase(), "TTKiroku.cgi");
      URLConnection  urlCon = url.openConnection();
      urlCon.setDoOutput(true);
      urlCon.setDoInput(true);
      urlCon.setAllowUserInteraction(false);
      PrintWriter writer = new PrintWriter(urlCon.getOutputStream());
      writer.print(message);
      writer.close();
      BufferedReader mReader = new BufferedReader(
                                  new InputStreamReader(
                                   urlCon.getInputStream(), "SJIS"));
      String url1=mReader.readLine();   // CGIから送られてきた URL
      while (mReader.readLine()!=null) {}  // あとは読み捨て
      mReader.close();
      getAppletContext().showDocument(new URL(getCodeBase(), url1));
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

run メソッドの後半部分である。ゲーム終了後、TTKiroku.cgi に結果を送る部分である。TTKiroku.cgi は、出力として URL を送ってくる。最後の

```
getAppletContext().showDocument(new URL(getCodeBase(), url1));
```

は、その URL に表示を切り替えている。

CGI側のプログラム（TTKiroku.java）は、送られてきたデータを表の形にして、tmp/TTKoroku.html というファイルに出力する。

ファイル TTKiroku.java（その 1）

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

class TTKiroku {
  static public void main(String[] args) {
    try {
      String cl = System.getProperty("CONTENT_LENGTH");
      int len = Integer.parseInt(cl);
      byte[] buf = new byte[len];
      System.in.read(buf);
      String answer = new String(buf, "ASCII");
      StringTokenizer st = new StringTokenizer(answer, "&");
```

（その 1）の部分は送られたデータを answer という変数に代入して、StringTokenizer で &で区切られたデータ毎に読む準備をしている。

ファイル TTKiroku.java（その 2）

```
      String fn = "tmp/TTKiroku.html";
      File f = new File(fn);
      PrintWriter fout = new PrintWriter
                  (new OutputStreamWriter(new FileOutputStream(f), "SJIS"));
      fout.println("<HTML><HEAD><TITLE>只今の結果</TITLE></HEAD><BODY>");
      fout.println("<H1 ALIGN=¥"CENTER¥">只今の結果</H1>");
      fout.println("<HR>");
      fout.println("<TABLE BORDER>");

      while(st.hasMoreTokens()) {
        String next = st.nextToken();
        fout.print("<TR>");
        StringTokenizer st1 = new StringTokenizer(next, "=");
        while(st1.hasMoreTokens()) {
          fout.print("<TD>"+st1.nextToken()+"</TD>");
        }
        fout.println("</TR>");
      }
      fout.println("</TABLE>");
      fout.println("</BODY></HTML>");
      fout.close();
```

（その 2）の部分では tmp/TTKiroku.html というファイルを開いて、TABLE タグを用いて記録の表を作っている。

ファイル TTKiroku.java（その 3）

```
      PrintWriter stdout =
          new PrintWriter(new OutputStreamWriter(System.out, "SJIS"));
      stdout.println("Content-type: text/plain");
      stdout.println();
      stdout.print(fn);
      stdout.close();

    } catch (Exception e) {}
  }
}
```

記録の表の書き込まれたファイルの URL をアプレット側に送り返している。これまで使っていた「Content-type: text/html」ではなく、「Content-type: text/plain」になっていることに注

意する。CGIからアプレットに送っているのは、HTMLのデータではなく、単なる文字列だからである。

問 3.2.1 （スコアランキングの表示）
上の例題で最後に自分の名前を入力してサーバに送ることができるように、アプレット側を改造せよ。サーバ側では送られたデータを記録しておき、スコアの順に記録を表示するようにせよ。

問 3.2.2 （ゲームの記録）
オセロや五目ならべのゲームのアプレットを作成し、そのゲームの経過をサーバに送信して記録出来るようにせよ。
（参考） ファイル *Othello.java*（超簡易版オセロゲーム）

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;

public class Othello extends Applet implements MouseListener {
  int scale = 40;
  int space = 3;
  int size = 8;
  int[][] state;

  public void init() {
    state = new int[8][8];
    addMouseListener(this);
  }

  public void paint(Graphics g) {
    int i,j;

    for (i=0; i<8; i++) {
      for (j=0; j<8; j++) {
        g.setColor(Color.green);
        g.fillRect(i*scale, j*scale, scale, scale);
        g.setColor(Color.black);
        g.drawRect(i*scale, j*scale, scale, scale);

        if (state[i][j]==1) {
          g.setColor(Color.white);
          g.fillOval(i*scale+space, j*scale+space, scale-space*2, scale-space*2);
        } else if (state[i][j]==2) {
          g.setColor(Color.black);
          g.fillOval(i*scale+space, j*scale+space, scale-space*2, scale-space*2);
        }
      }
    }
  }

  public void mouseClicked(MouseEvent me) {
    int x = me.getX()/scale; int y = me.getY()/scale;
    int mod = me.getModifiers();
    if (0<=x && x<8 && 0<=y && y<8) {
       if ((mod & MouseEvent.BUTTON1_MASK) != 0)        // 左ボタン
         state[x][y] = (state[x][y]+1)%3;
       else if ((mod & MouseEvent.BUTTON3_MASK) != 0)   // 右ボタン
         state[x][y] = (state[x][y]+2)%3;
    }
    repaint();
  }

  public void mouseEntered(MouseEvent me) {}
  public void mouseExited(MouseEvent me) {}
  public void mousePressed(MouseEvent me) {}
  public void mouseReleased(MouseEvent me) {}
}
```