第3章　GETとPOSTによるパラメータ渡し

3.1　Servletへのパラメータ渡し（GET編）

これまで紹介した例はブラウザ側から Servlet にデータを渡すことはなかったが、ブラウザから Servlet にパラメータを渡して Servlet の振舞いを変えることも可能である。CGI/Servlet などのサーバサイドプログラムにパラメータを渡す方法には GET と POST の 2 種類がある。ここではまず GET について説明する。

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 Servlet（HiLite.java）を作成する。

ファイル HiLite.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class HiLite extends HttpServlet {
  @Override
  public void doGet(HttpServletRequest request, HttpServletResponse response)
              throws IOException {
    response.setContentType("text/html; charset=Windows-31J");
    PrintWriter out = response.getWriter();
    out.println("<html><head></head><body><pre>");
    // HiLite.java のコピーを WEB アプリのルートフォルダに置いておくこと
    File f = new File(getServletContext().getRealPath("/HiLite.java"));
    String word = request.getQueryString();
    InputStreamReader fr = new InputStreamReader
                               (new FileInputStream(f), "Windows-31J");
    BufferedReader in = new BufferedReader(fr);

    while(true) {
      String line = in.readLine();
      if (line==null) break;
      line = line.replace("&", "&amp;");
      line = line.replace("<", "&lt;");
      line = line.replace(">", "&gt;");
      if (word!=null && word.length()!=0) {
          line = line.replace(word, "<font color='red'>"+word+"</font>");
      }
      out.println(line);
    }
    out.println("</pre></body></html>");
    out.close();
  }
}
```

GET で Servlet 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（Query String と呼ぶ）がパラメータとして Servlet に渡される。例えば

```
http://localhost:8080/SoftEngEnshu/HiLite?print
```

の、print の部分が Query String（パラメータ）になる。

Servlet プログラム中では、このパラメータは doGet の第 1 引数（HttpServletRequest クラス）に対する getQueryString() というメソッドで受け取ることができる。結局、

```
String word = request.getQueryString();
```

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 = line.replace(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、replace 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。

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、表示が乱れるおそれがある。次の部分

```
line = line.replace("&", "&amp;");
line = line.replace("<", "&lt;");
line = line.replace(">", "&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（これは別のファイルにしても良い）の中のキーワードを赤色で表示することになる。

```
http://localhost:8080/SoftEngEnshu/HiLite?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。

問 3.1.1 （カレンダー）
例えば、*MyCalerdar?200410* のような形でパラメータを渡されると、*2004* 年 *10* 月のカレンダーを作成するような *Servlet* 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 3.1.2 （スライドショー）
images 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 *Servlet*（*SlideShow.java*）を作成せよ。

ヒント: パラメータが渡されなかった場合（request.getQueryString() の戻り値が *null* になる）は、*1.png* を表示する。パラメータが *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='images/n.png' /><hr/>
<a href='SlideShow?n-1'>前</a>
<a href='SlideShow?n+1'>次</a>
</div>
</body></html>
```

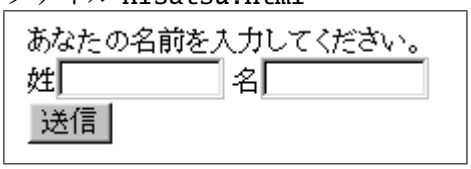
ただし、*SlideShow* は設置するサーブレット自身の名前である。

3.2 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='servlet/Aisatsu' 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='*URI*' method='post'>〜</form>

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

なお、method='post' ではなく、method='get' とすると、以前に紹介した URI の最後に?をつけてパラメータを渡す方式（GET）で CGI/Servlet 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='*namae*' />

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、*namae* 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='*namae*' 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/Servlet 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='*namae*' 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、*namae* 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/Servlet 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='*namae*' value='*str*' />**
隠し要素（hidden）は画面には表示されないが、名前と値は CGI/Servlet 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/Servlet 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='*namae*'>〜</textarea>**
複数行の文字が入力できるテキストボックスを表示する。*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。

3.3 Servlet へのパラメータ渡し（POST 編）

POST でデータを受け取る場合、Servlet は doGet の代わりに doPost というメソッドを実行するので、この doPost メソッドを定義する必要がある。

実はフォームからデータは次のような形の文字列として送られる。

$$name_1{=}value_1\&name_2{=}value_2\& \ldots \ \&name_n{=}value_n$$

$name_1$, $name_2$, ... は input タグや textarea タグに付けられていた name 属性で $value_1$, $value_2$, ... 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。

この value 属性を Servlet 中で読み込むには doPost の第 1 引数（HttpServletRequest クラス）に対する getParameter メソッドを使う。getParameter メソッドの引数は name 属性を指定する。getParameter メソッドは上のようなフォームから送られてくるデータを解析して対応する値（value 属性）を返す。

なお、フォームに日本語を入力する場合、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（元の文字コードが Windows-31J 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 Servlet である。（この例では Aisatsu.html はアプリケーションルートの直下に置かれると仮定している。）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。
ファイル Aisatsu.java

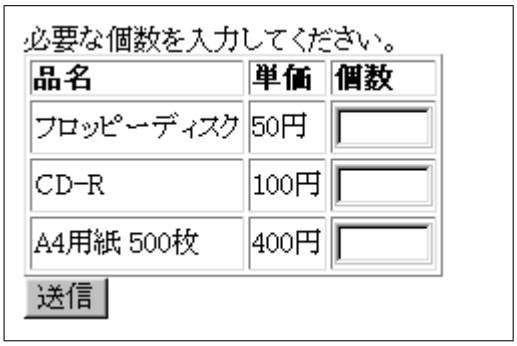
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Aisatsu extends HttpServlet {
  @Override
  public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws IOException {
    response.setContentType("text/html; charset=Windows-31J");
    PrintWriter out = response.getWriter();
    out.println("<html><head></head><body>");
    request.setCharacterEncoding("Windows-31J");
    String family = request.getParameter("family");
    String given  = request.getParameter("given");
    out.println("こんにちは,"+family+" "+given+"!");
    out.println("</body></html>");
    out.close();
  }
}
```

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。

```
request.setCharacterEncoding("Windows-31J");
```

の setCharacterEncoding メソッドはフォームに使われている文字コードを指定する[1]。”Windows-31J”の代わりに”JISAutoDetect”とすると、 Shift-JIS, EUC-JP, ISO 2022 JP のなかから自動判別する。（ただし文字列が短い場合は自動判別を間違う場合があるのであまりお奨めできない。）

問 3.3.1 （見積り作成 *Servlet*）
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 *Servlet* 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。）

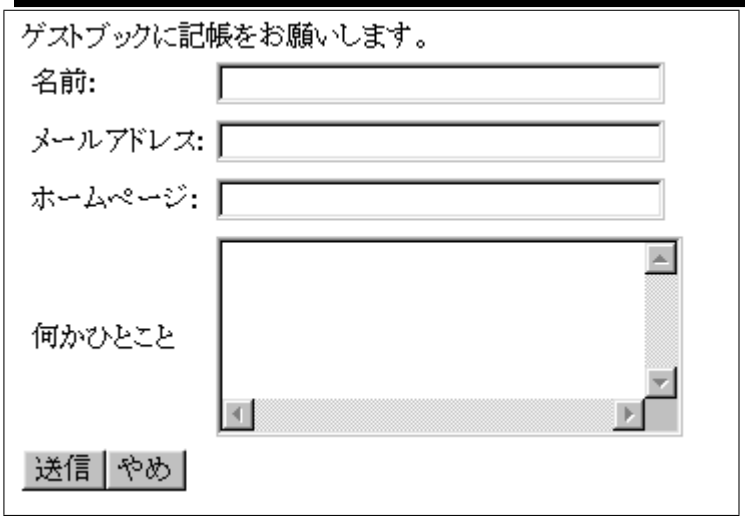


[1]通常の行儀の良いブラウザの場合は、フォームが書かれていた HTML ファイル（上の例の場合は Aisatsu.html）の文字コードと同じ文字コードでエンコーディングしてくれる。

問 3.3.2 （*HiLite* 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 *Servlet* を作成せよ。

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 Servlet である。フォームの HTML のソースはつぎのようになる。（この例ではアプリケーションルートに作成すると仮定する。）
ファイル GuestBook.html

```
<html><head><title>ゲストブック記帳</title></head>
<body>
<form action='servlet/GuestBook' 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>
```



Servlet 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アプリケーションルートに作成しておく。

ファイル Guests.html

```
<?xml version="1.0" encoding="Shift_JIS"?>
<html xmlns="http://www.w3.org/1999/xhtml">
<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割して提示する。

ファイル GuestBook.java（その 1）

```
import java.io.*;
import java.util.*;   // Enumeration 用に必要
import javax.servlet.*;
import javax.servlet.http.*;

public class GuestBook extends HttpServlet {
```

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）

```
  @Override
  public void doPost(HttpServletRequest request, HttpServletResponse response)
              throws IOException {
    response.setContentType("text/html; charset=Windows-31J");
    request.setCharacterEncoding("Windows-31J");
    PrintWriter out = response.getWriter();
    out.println("<html><head></head><body>");

    File f = new File(getServletContext().getRealPath("/Guests.html"));
    BufferedReader fin = new BufferedReader
                (new InputStreamReader(new FileInputStream(f), "Windows-31J"));

    File tmp = new File(getServletContext().getRealPath("/tmp.html"));
    PrintWriter fout = new PrintWriter
              (new OutputStreamWriter(new FileOutputStream(tmp), "Windows-31J"));
    while (true) {
      String line = fin.readLine();
      if (line.trim().equals("<!-- OWARI -->")) break;
      fout.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）

```
    fout.println("<table border>");
    Enumeration keys = request.getParameterNames();
    while (keys.hasMoreElements()) {
      fout.print("<tr>");
      String left  = (String)keys.nextElement();
      String right = request.getParameter(left);
      fout.print("<td>"+left+"</td>");
      fout.print("<td>"+right+"</td>");
      fout.print("</tr>");
    }
    fout.println("</table>");
    fout.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。getParameterNames メソッドはフォームの中で使われているすべての name 属性を列挙したデータ（Enumeration 型）を返す。Enumeration クラスは何かの “集まり” を表すデータ型である。このクラスのオブジェクトに対しては、hasMoreElements というメソッドでデータがまだ残っているかどうかを調べることができる。データが残っている場合は nextElement というメソッドで次のデータを取り出すことができる。ただし nextElement メソッドの戻り値の型は Object 型という一般的な型なので、このプログラムの場合は String 型にキャストする必要がある。

Enumeration については、Java の API ドキュメント:

http://java.sun.com/j2se/1.5.0/ja/docs/ja/api/java/util/Enumeration.html

も参照すること。

ファイル GuestBook.java（その 4）

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

（その 4）では、次に Servlet 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。最後に両方のストリームを close() して、Guests.html を削除し、tmp.html の名前を Guests.html に変更している。

ファイル GuestBook.java（その 5）

```
    out.println("御記帳有難うございました。<br>");
    out.print("ゲストブックは<a href='Guests.html'>こちら</a>です。");

    out.println("</body></html>");
    out.close();
  }
}
```

お礼の文章と、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 3.3.3 （日記作成 *Servlet*）
「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 *Servlet*（`Diary.java`）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 3.3.4 （家計簿）
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 *Servlet* を作成せよ。入力した項目に加えて、その日の支出の計とそれまでの支出の累計の両方を計算してテーブルの形に整形し、ゲストブックとおなじようにファイルにテーブルを追加していくようにせよ。

3.4 参考: DOM の利用

Guests.html のような HTML ファイル（一般に XML ファイル）を操作するとき、単なるテキスト形式ではなく、木構造として操作できると便利である。XML を木構造として扱うための取り決めとして DOM（Document Object Model）がある。

以下に参考までに `GuestBook.java` とほぼ同等の処理を DOM を用いて実現したプログラムを掲載する。個々のクラスやメソッドについては、Java の API ドキュメント（`http://java.sun.com/j2se/1.5.0/ja/docs/ja/api/`）に掲載されている説明を参考に、各自で調べて欲しい。

ファイル `GuestBookDom.java`（import 宣言部分）

```
import java.io.*;
import java.util.*;
import javax.servlet.http.*;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;

import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Text;
```

ファイル GuestBookDom.java（クラスメソッド部分）

```
public class GuestBookDom extends HttpServlet {

  private static void appendNewline(Document doc, Element e) {
    Text txt = doc.createTextNode("¥n");
    e.appendChild(txt);
  }

  private static Element createTableFromKeys(HttpServletRequest request,
                                Document doc, Enumeration keys) {
    Element tbl = doc.createElement("table");
    tbl.setAttribute("border", "1");
    appendNewline(doc, tbl);
    while (keys.hasMoreElements()) {
      Element row = doc.createElement("tr");
      Element td1 = doc.createElement("td");
      String left = (String)keys.nextElement();
      td1.setTextContent(left);
      row.appendChild(td1);
      Element td2 = doc.createElement("td");
      td2.setTextContent(request.getParameter(left));
      row.appendChild(td2);
      tbl.appendChild(row);
      appendNewline(doc, tbl);
    }
    return tbl;
  }
```

ファイル GuestBookDom.java（doPost メソッド）

```
  @Override
  public void doPost(HttpServletRequest request, HttpServletResponse response)
  throws IOException {
    response.setContentType("text/html;  charset=Windows-31J");
    request.setCharacterEncoding("Windows-31J");
    PrintWriter out = response.getWriter();
    out.println("<html><head></head><body>");
    try {
      // 読み込みの準備
      String filename = getServletContext().getRealPath("/Guests.html");
      DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
      DocumentBuilder db = dbf.newDocumentBuilder();

      Document doc = db.parse(filename);       // 木構造の生成
      Element root = doc.getDocumentElement(); // root は html 要素のはず
      Element body = (Element)root.getElementsByTagName("body").item(0);
                     // body 要素の検索

      // 新しい table の追加
      body.appendChild(createTableFromKeys(request, doc,
                                           request.getParameterNames()));
      appendNewline(doc, body);
      body.appendChild(doc.createElement("hr"));
      appendNewline(doc, body);

      // 出力処理
      TransformerFactory fac = TransformerFactory.newInstance();
      Transformer tran = fac.newTransformer();
      tran.setOutputProperty(OutputKeys.ENCODING, "Shift_JIS");
                              //  注: Windows-31J は少し不都合がある
      OutputStream o = new FileOutputStream(filename);
      StreamResult result = new StreamResult(o);
      tran.transform(new DOMSource(doc.getDocumentElement()), result);
      o.close();

      out.println("御記帳有難うございました。<br>");
      out.println("ゲストブックは<a href='Guests.html'>こちら</a>です。");

    } catch (Exception e) {
      e.printStackTrace(out);
    }
    out.println("</body></html>");
    out.close();
  }
}
```

3.5 参考: デバッグ用サーブレット

POST を利用する Servlet 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と便利である。それには次のような Servlet を利用すれば良い。

ファイル Debug.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;

public class Debug extends HttpServlet {
  @Override
  public void doPost(HttpServletRequest request, HttpServletResponse response)
              throws IOException {
    response.setContentType("text/html; charset=Windows-31J");
    PrintWriter out = response.getWriter();
    out.println("<html><head></head><body>");
    int len = request.getContentLength();
    char[] buf = new char[len];
    File f = new File(getServletContext().getRealPath("/Debug.out"));
    PrintWriter fout = new PrintWriter(new FileWriter(f));
    BufferedReader in = request.getReader();
    in.read(buf);
    in.close();
    fout.print(buf);
    fout.close();
    out.println("<tt>CONTENT_LENGTH</tt>は "+len+"です。");
    out.println("<a href='Debug.out'>Debug.out</a>に情報を書きこみました。");
    out.println("</body></html>");
    out.close();
  }
}
```

この Servlet 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す。

キーワード:

GET, Query String, getParameter メソッド, フォーム, POST, doPost メソッド, setCharacterEncoding メソッド, getParameterNames メソッド, Enumeration クラス, DOM