第2章　GETとPOST

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 にパラメーターを渡して Servlet の振舞いを変えることも可能である。CGI/Servlet などのサーバーサイドプログラムにパラメーターを渡す方法には GET と POST の 2 種類がある。

GET は URL の後ろに '?' という文字をつけ、その後にパラメーターを書く方法で、FORM を用意しなくても、簡易にパラメーターを渡すことができる。その代わり、送ることのできるデータのバイト数に制限がある。

GET によるパラメーターの例:

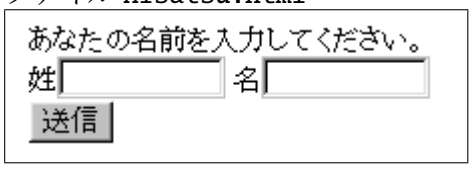
```
http://maps.google.co.jp/maps?hl=ja&ll=34.292821,134.063587&z=15
```

POST は HTML のフォームからデータを送る必要がある。送ることのできるデータのバイト数に制限はない。

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



HTML のソース

```
1 <form action='Aisatsu' method='post'>
2 あなたの名前を入力してください。<br/>
3 姓<input type='text' size='10' name='family' />
4 名<input type='text' size='10' name='given' />
5 <br/>
6 <input type='submit' value='送信' />
7 </form>
```

form タグの action 属性にデータを受けとるサーブレット（一般にサーバーサイドプログラム）の URL を指定する。この例では、同じ階層にある、Aisatsu というサーブレットにデータを送る。

2.1　Servlet へのパラメーター渡し（GET 編）

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ーとして受け取り、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（HighLight.java）を作成する。

ファイル HighLight.java

```
1  import java.io.BufferedReader;
2  import java.io.File;
3  import java.io.FileInputStream;
4  import java.io.IOException;
5  import java.io.InputStreamReader;
6  import java.io.PrintWriter;
7
8  import javax.servlet.http.HttpServlet;
9  import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 public class HighLight extends HttpServlet {
13   @Override
14   public void doGet(HttpServletRequest request,
15                     HttpServletResponse response)
16               throws IOException {
17     response.setContentType("text/html;␣charset=Windows-31J");
18     PrintWriter out = response.getWriter();
19     out.println("<html><head></head><body><pre>");
20     // 適当な Java のソースファイル（例えば HighLight.java のコピー）を
21     // WEB アプリのルートフォルダに Tekito.java という名前で
22     // 置いておくこと
23     File f = new File(getServletContext()
24                       .getRealPath("/Tekito.java"));
25     String word = request.getQueryString();
26     InputStreamReader fr
27      = new InputStreamReader(new FileInputStream(f), "Windows-31J");
28     BufferedReader in = new BufferedReader(fr);
29
30     while(true) {
31       String line = in.readLine();
32       if (line==null) break;
33       line = line.replace("&", "&amp;");
34       line = line.replace("<", "&lt;")
35       line = line.replace(">", "&gt;");
36
37       if (word!=null && word.length()!=0) {
38         line = line.replace(word,
39                             "<font␣color='red'>"+word+"</font>");
40       }
```

```
41        out.println(line);
42      }
43      out.println("</pre></body></html>");
44      out.close();
45    }
46  }
```

GET で Servlet にパラメーターを渡すには 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）がパラメーターとして Servlet に渡される。例えば

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

の、print の部分が Query String（パラメーター）になる。

Servlet プログラム中では、このパラメーターは 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 HighLight.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/HighLight?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 2.1.1 カレンダー

例えば、MyCalerdar?200804 のような形でパラメーターを渡されると、2008 年 4 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、java.util.Calendar クラスのメソッドもしくは次のような Zellar の公式を用いよ

```
  static int Zellar(int y, int m, int d) { // 0 が日曜、1 が月曜、... 6が土曜
    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;
  }
```

問 2.1.2 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 : パラメーター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。パラメーターが *n* のときは、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。

2.2　フォーム

この節では、HTML のフォーム（Form）の書き方を説明する。

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 や 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ーを渡す方式（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。

2.3 Servlet へのパラメーター渡し（POST 編）

POST でデータを受け取る場合（つまり、フォームからデータを受け取る場合の大半）、Servlet は doGet ではなく doPost というメソッドを実行する。Servlet では、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。

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

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

例題: おうむ返し
この章の最初に例として紹介した 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。）Aisatsu.html の「送信」ボタンを押すと、その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

```
1  import java.io.IOException;
2  import java.io.PrintWriter;
3
4  import javax.servlet.http.HttpServlet;
5  import javax.servlet.http.HttpServletRequest;
6  import javax.servlet.http.HttpServletResponse;
7
8  public class Aisatsu extends HttpServlet {
9    @Override
10   public void doPost(HttpServletRequest request,
11                      HttpServletResponse response)
12             throws IOException {
13     response.setContentType("text/html;␣charset=Windows-31J");
14     PrintWriter out = response.getWriter();
15     out.println("<html><head></head><body>");
16     request.setCharacterEncoding("Windows-31J");
17     String family = request.getParameter("family");
18     String given  = request.getParameter("given");
19     out.printf("こんにちは,%s␣%s!%n", family, given);
20     out.println("</body></html>");
21     out.close();
22   }
23 }
```

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。そのためには、

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

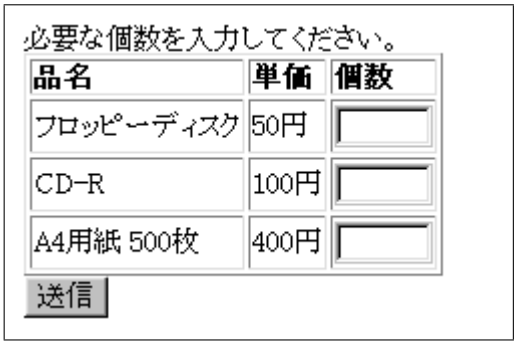
の setCharacterEncoding メソッドで、フォームに使われている文字コードを指定する[1]。”Windows-31J”の代わりに”JISAutoDetect”とすると、 Shift_JIS, EUC-JP,

[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。

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。）

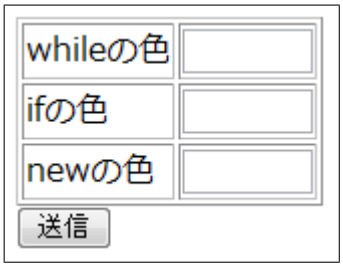
問 2.3.1 見積り作成 Servlet

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。）



問 2.3.2 HighLight の色を入力

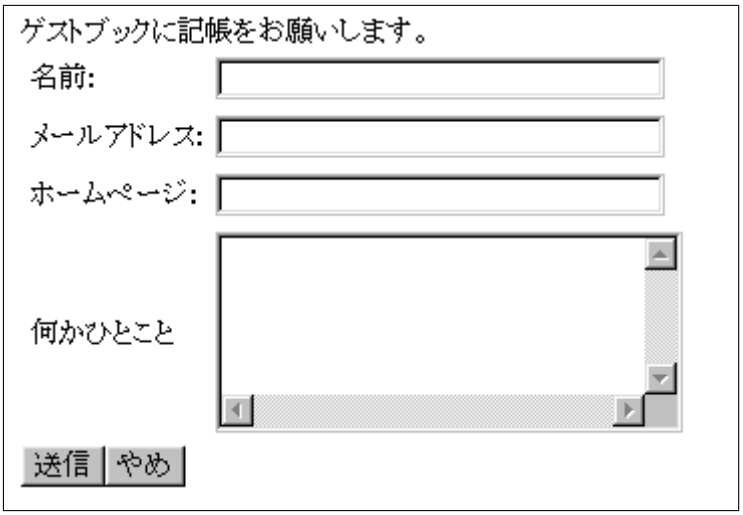
右のようなフォームから入力を受け取って、あるファイル中の指定された文字列を、フォームで入力された色で強調して表示する Servlet を作成せよ。



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 は次のような内容でアプリケーションルートに作成する。）
ファイル GuestBook.html

```
1  <html><head>
2  <meta http-equiv='Content-Type'
3        content='text/html; charset=Windows-31J'>
4  <title>ゲストブック記帳</title></head>
5  <body>
6  <form action='GuestBook' method='post'>
7  ゲストブックに記帳をお願いします。<br/>
8  <table>
9  <tr><td>名前:</td>
10     <td><input type='text' size='30' name='名前'></td></tr>
11 <tr><td>メールアドレス:</td>
12     <td><input type='text' size='30' name='メールアドレス'></td>
13 </tr>
14 <tr><td>ホームページ:</td>
15     <td><input type='text' size='30' name='ホームページ'></td>
16 </tr>
17 <tr><td>何かひとこと</td>
18     <td><textarea name='ひとこと' rows='5' cols='30'></textarea>
```

```
19          </td></tr>
20  </table>
21  <input type='submit' value='送信'>
22  <input type='reset' value='やめ'>
23  </form>
24  </body></html>
```



Servlet では、Guests.html というファイルの最後のほう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 /WEB-INF に作成しておく。

ファイル Guests.html

```
1  <?xml version="1.0" encoding="Windows-31J"?>
2  <html>
3  <head><title>ゲストブック</title></head>
4  <body>
5  <h1 align="center">ゲストブック</h1>
6  御記帳有難うございました。<br/>
7  <hr/>
8  </body>
9  </html>
```

プログラムは長いのでいくつかに分割して提示する。最初はこれまでのプログラムとあまり違う点はない。

ファイル GuestBook.java（その 1）

```
1  import java.io.BufferedReader;
2  import java.io.File;
3  import java.io.FileInputStream;
4  import java.io.FileNotFoundException;
```

```
5  import java.io.FileOutputStream;
6  import java.io.IOException;
7  import java.io.InputStreamReader;
8  import java.io.OutputStreamWriter;
9  import java.io.PrintWriter;
10
11 import javax.servlet.ServletException;
12 import javax.servlet.http.HttpServlet;
13 import javax.servlet.http.HttpServletRequest;
14 import javax.servlet.http.HttpServletResponse;
15
16 public class GuestBook extends HttpServlet {
```

（その 2）では、Guests.html，tmp.html の 2 つのファイルをオープンし、“</body>” 文字列を含む行が現れるまでは、単に Guests.html から tmp.html へコピーをする。

ファイル GuestBook.java（その 2）

```
17   @Override
18   public void doPost(HttpServletRequest request,
19                      HttpServletResponse response)
20               throws IOException {
21     request.setCharacterEncoding("Windows-31J");
22     File tmp = new File(getServletContext()
23                         .getRealPath("/WEB-INF/tmp.html"));
24     PrintWriter fout = new PrintWriter
25            (new OutputStreamWriter(new FileOutputStream(tmp),
26                                    "Windows-31J"));
27     File f = new File(getServletContext()
28                       .getRealPath("/WEB-INF/Guests.html"));
29     try {
30       BufferedReader fin = new BufferedReader
31              (new InputStreamReader(new FileInputStream(f),
32                                     "Windows-31J"));
33       String line;
34       while (true) {
35         line = fin.readLine();
36         if (line==null) {
37           fout.println("<!--␣内部エラー:␣"
38                        + "Guests.htmlが壊れています。-->");
39           line="</body>";
40           break;
41         }
42         if (line.contains("</body>")) break;
43         fout.println(line);
44       }
```

次に（その 3）では、フォームから入力されたデータからテーブルを生成している。

ファイル GuestBook.java（その 3）

```
45      fout.println("<table␣border>");
46      fout.printf("<tr><td>名前</td><td>%s</td></tr>%n",
47                  request.getParameter("名前"));
48      fout.printf("<tr><td>メールアドレス</td><td>%s</td></tr>%n",
49                  request.getParameter("メールアドレス"));
50      fout.printf("<tr><td>ホームページ</td><td>%s</td></tr>%n",
51                  request.getParameter("ホームページ"));
52      fout.printf("<tr><td>ひとこと</td><td>%s</td></tr>%n",
53                  request.getParameter("ひとこと"));
54      fout.println("</table>");
55      fout.println("<hr/>");
```

（その 4）では、line（その 2 で読み込まれていた</body>を含む行）を出力し、Guests.html の残りの部分を tmp.html にコピーする。両方のストリームを close() する。

ファイル GuestBook.java（その 4）

```
56      fout.println(line);
57      while (true) {
58        line = fin.readLine();
59        if (line==null) break;
60        fout.println(line);
61      }
62      fin.close();
63      fout.close();
64    } catch (FileNotFoundException e) {
65      outputError(response, e);
66      return;
67    }
```

（その 5）では、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）

```
68    f.delete();
69    tmp.renameTo(f);
```

（その 6）では、Guest.html の中身を表示するように処理をフォワード（転送）する。

まず ServletContext の getRequestDispatcher というメソッドの引数にフォワード先のパスを指定して RequestDispatcher オブジェクトを取り出す。このパスは /から始まらなくてはならず、現在の Web アプリケーションのコンテキストルートからの相対パスと解釈される。次に RequestDispatcher オブジェクトに対して request と response を引数として forward メソッドを呼び出す。

WEB-INF フォルダの下は、ブラウザから直接リクエストされたときにはアクセスが拒否されるようになっているが、forward メソッド（や include メソッド）で間接的にリクエストされたときはアクセスできるようになっている。

ファイル GuestBook.java（その 6）

```
70    try {
```

```
71        getServletContext()
72          .getRequestDispatcher("/WEB-INF/Guests.html")
73          .forward(request, response);
74      } catch (ServletException e) {
75        outputError(response, e);
76        return;
77      }
78    }
79
80    private void outputError(HttpServletResponse response,
81                             Exception e)
82            throws IOException {
83      response.setContentType("text/html;␣charset=Windows-31J");
84      PrintWriter out = response.getWriter();
85      out.println("<html><body><pre>");
86      e.printStackTrace(out);
87      out.println("</pre></body></html>");
88    }
89  }
```

問 2.3.3 ゲストブックを、単一の HTML ファイルに記録する方式ではなく、投稿ごとに個別のファイルに保存する方式に変更せよ。

問 2.3.4 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 2.3.5 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。

2.4　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。

DOM については、Java の javax.xml.parsers パッケージ、javax.xml.transform パッケージ、org.w3c.dom パッケージの API ドキュメントの説明を参考に、各自で調べて欲しい。

ファイル GuestBookDom.java

```
1  import java.io.FileOutputStream;
2  import java.io.FileReader;
3  import java.io.IOException;
4  import java.io.OutputStream;
5  import java.io.PrintWriter;
6  import java.util.Enumeration;
7
8  import javax.servlet.http.HttpServlet;
9  import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11 import javax.xml.parsers.DocumentBuilder;
12 import javax.xml.parsers.DocumentBuilderFactory;
13 import javax.xml.transform.OutputKeys;
14 import javax.xml.transform.Transformer;
15 import javax.xml.transform.TransformerFactory;
16 import javax.xml.transform.dom.DOMSource;
17 import javax.xml.transform.stream.StreamResult;
18
19 import org.w3c.dom.Document;
20 import org.w3c.dom.Element;
21
22 public class GuestBookDom extends HttpServlet {
23   private static
24     Element createTableFromKeys(HttpServletRequest request,
25                                 Document doc, Enumeration keys) {
26     Element tbl = doc.createElement("table");
27     tbl.setAttribute("border", "1");
28     tbl.appendChild(doc.createTextNode("{\yen}n"));
29     while (keys.hasMoreElements()) {
30       Element row = doc.createElement("tr");
31       Element td1 = doc.createElement("td");
32       String left = (String)keys.nextElement();
33       td1.setTextContent(left);
34       row.appendChild(td1);
35       Element td2 = doc.createElement("td");
36       td2.setTextContent(request.getParameter(left));
37       row.appendChild(td2);
38       tbl.appendChild(row);
39       tbl.appendChild(doc.createTextNode("{\yen}n"));
```

```
40      }
41      return tbl;
42    }
43
44    @Override
45    public void doPost(HttpServletRequest request,
46                       HttpServletResponse response)
47    throws IOException {
48      request.setCharacterEncoding("Windows-31J");
49      try {
50        // 読み込みの準備
51        String filename = getServletContext()
52                          .getRealPath("/WEB-INF/Guests.html");
53        DocumentBuilderFactory dbf
54          = DocumentBuilderFactory.newInstance();
55        DocumentBuilder db = dbf.newDocumentBuilder();
56
57        Document doc = db.parse(filename);       // 木構造の生成
58        // root は html 要素のはず
59        Element root = doc.getDocumentElement();
60        Element body = (Element)root.getElementsByTagName("body")
61                                    .item(0);
62                    // body 要素の検索
63
64        // 新しい table の追加
65        body.appendChild(createTableFromKeys(request, doc,
66                                       request.getParameterNames()));
67        body.appendChild(doc.createTextNode("{\yen}n"));
68        body.appendChild(doc.createElement("hr"));
69        body.appendChild(doc.createTextNode("{\yen}n"));
70
71        // 出力処理
72        TransformerFactory fac = TransformerFactory.newInstance();
73        Transformer tran = fac.newTransformer();
74        tran.setOutputProperty(OutputKeys.ENCODING, "Shift_JIS");
75                              // 注: Windows-31J は少し不都合がある
76        OutputStream o = new FileOutputStream(filename);
77        StreamResult result = new StreamResult(o);
78        tran.transform(new DOMSource(doc.getDocumentElement()),
79                                       result);
80        o.close();
81        getServletContext()
82          .getRequestDispatcher("/WEB-INF/Guests.html")
83          .forward(request, response);
84
85      } catch (Exception e) {
86        response.setContentType("text/html;␣charset=Windows-31J");
87        PrintWriter out = response.getWriter();
88        out.println("<html><body><pre>");
```

```
89        e.printStackTrace(out);
90        out.println("</pre></body></html>");
91      }
92    }
93  }
```

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 , getRequestDispatcher メソッド , forward メソッド , DOM