第II章 GET と POST

これまで紹介した例はブラウザー側から Servlet にデータを渡すことはなかったが、ブラウザーから 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、フォームを用意しなくても、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。

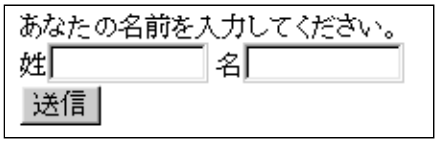
> GETによるパラメーターの例:
> `http://maps.google.co.jp/maps?hl=ja&ll=34.292821,134.063587&z=15`

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

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>簡単な Form</title>
</head>
<body>
<form action='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>
</body>
</html>
```



> 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。

II.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: Query String のオウム返し

まずは、パラメーターをオウム返しするプログラムである。

ファイル QueryStringTest.java

```java
import java.io.IOException;
import java.io.PrintWriter;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;


@WebServlet("/QueryStringTest")
public class QueryStringTest extends HttpServlet {
    private static final long serialVersionUID = 1L;
    @Override
    protected void doGet(HttpServletRequest request,
                         HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType(
            "text/html; charset=UTF-8");
        PrintWriter out = response.getWriter();
        out.println("<html><head></head><body>");
        String qs = request.getQueryString();

        out.printf("QueryString は  %s です。%n", qs);
        out.println("</body></html>");
        out.close();
    }
}
```

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/OOPL/QueryStringTest?hello
```

の、`hello` 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 qs = request.getQueryString();
```

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

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サーブレット（`HighLight.java`）を作成する。

ファイル HighLight.java

```java
import java.io.BufferedReader;
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;


@WebServlet("/HighLight")
public class HighLight extends HttpServlet {
  @Override
  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType(
      "text/html; charset=UTF-8");
    PrintWriter out = response.getWriter();
    out.println("<html><head></head><body><pre>");
    // 適当な Java のソースファイル
    // （例えば HighLight.java のコピー）
    // を WEB アプリのルートフォルダに Tekito.txt
    // という名前で置いておくこと
    File f = new File(getServletContext()
                      .getRealPath("/Tekito.txt"));
    String word = request.getQueryString();
    InputStreamReader fr = new InputStreamReader
            (new FileInputStream(f), "UTF-8");
    BufferedReader in = new BufferedReader(fr);

    while(true) {
      String line = in.readLine();
      if (line == null) break;
      line = line.replace("&", "&");
      line = line.replace("<", "<");
      line = line.replace(">", ">");

      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();
    in.close();
 }
}
```

このプログラムは、Query String を word という変数に入れ、

```java
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 II.1.1 カレンダー

例えば、MyCalerdar?202004 のような形でパラメーターを渡されると、2020 年 4 月のカレンダーを作成するようなサーブレットMyCalendar.java 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

```
// 0 が日曜、1 が月曜、... 6 が土曜
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;
}
```

問 II.1.2 スライドショー

images ディレクトリー中に 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サーブレット（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（<u><em>n</em></u>）
</title></head>
<body>
<div align='center'>
<img src='images/<u><em>n</em></u>.png' /><hr/>
<a href='SlideShow?<u><em>n-1</em></u>'>前</a>
<a href='SlideShow?<u><em>n+1</em></u>'>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。

1.2 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。

II.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。

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

$name_1$, $name_2$, … 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。

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

```java
import java.io.IOException;
import java.io.PrintWriter;

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

@WebServlet("/Aisatsu")
public class Aisatsu extends HttpServlet {
```

```
12    @Override
13    protected void doGet(HttpServletRequest request,
14                         HttpServletResponse response)
15              throws ServletException ,IOException {
16      response.sendRedirect("Aisatsu.html");
17    }
18
19    @Override
20    protected void doPost(HttpServletRequest request,
21                          HttpServletResponse response)
22              throws ServletException, IOException {
23      response.setContentType(
24        "text/html; charset=UTF-8");
25      PrintWriter out = response.getWriter();
26      out.println("<html><head></head><body>");
27      request.setCharacterEncoding("UTF-8");
28      String family = request.getParameter("family");
29      String given  = request.getParameter("given");
30      out.printf("こんにちは, %s %s!%n", family, given);
31      out.println("</body></html>");
32      out.close();
33    }
34  }
```

Aisatsu.java では、主に POST で処理をするため、 doGet では、 Aisatsu.html の中身を表示するように処理をリダイレクトしている。

リダイレクトは、いったんブラウザー側に転送先の URL を送り、ブラウザーがこの URL に改めてリクエストを送る。これによって別のサーブレット（または JSP）が表示を分業する。HttpServletResponse クラスの sendRedirect というメソッド を用いる。 sendRedirect の引数は、転送先のURLである。

```
response.sendRedirect(URLString);
```

なお、詳しく説明しないが、同じ HttpServletResponse クラスの encodeRedirectURL メソッドと併用する場合もある。

```
response.sendRedirect(response.encodeRedirectURL(URLString));
```

doPostメソッドは、

```
request.setCharacterEncoding("UTF-8");
```

という呼出しで始まっている。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“%e9%a6%99%e5%b7%9d%e5%a4%a7%e5%ad%a6”とエンコードされる（元の文字コードが UTF-8 の場合）。そこでサーブレット側でこれをデコードする必要がある。そのために、 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している (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。)。"UTF-8" 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 II.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（右の上の図）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サーブレット `Mitsumori.java` を作成せよ。（さらに結果を見積書のようにテーブル（右の下の図）として整形せよ。）なお、単価などのデータは（本来はデータベースから取得するものだが）プログラム中に定数として埋め込んでおいてよい。

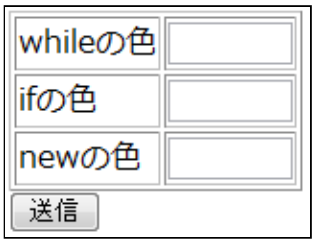
品名	単価	個数	小計
BD-Rディスク	500円	3	1500円
インクカートリッジ	2000円	2	4000円
A4用紙 500枚	400円	4	1600円
合計			7100円

なお、水平方向に複数個結合したセルを表示するには `td` タグの `colspan` 属性を使用する。

```
1 <tr><td colspan='3'>合計</td><td>7100円</td></tr>
```

問 II.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サーブレット `HighLight2.java` を作成せよ。



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,