第2章　ファイル・ディレクトリ操作

Servlet のようなサーバーサイドプログラムは、アクセスカウンタにせよ、掲示板にせよファイルやデータベースにアクセスする必要がある場合が多い。（でなければ、クライアントサイドのプログラムで実現できることがことが多い。）

この章では Java のファイルやディレクトリ操作の API を使用し、サーバーサイドでファイルアクセスを行なう Servlet を作成する。

2.1　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サーバサイドプログラムで、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

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

public class Counter extends HttpServlet {
  @Override
  public void doGet(HttpServletRequest request, HttpServletResponse response)
              throws IOException {

    int i;
    response.setContentType("text/html; charset=Windows-31J");
    PrintWriter out = response.getWriter();
    out.println("<html><head></head><body>");

    File f = new File(getServletContext().getRealPath("/counter.txt"));
    try {
      BufferedReader fin = new BufferedReader(new FileReader(f));
      i =  Integer.parseInt(fin.readLine());
      fin.close();  // close を忘れない
    } catch (FileNotFoundException e) {
      i = 0;        // ファイルがなければ 0 に
    }

    PrintWriter fout = new PrintWriter(new FileWriter(f));
    fout.println(++i);
    fout.close();   // close を忘れない

    out.printf("あなたは %d 番目の来訪者です。%n", i);
    out.println("</body></html>");
    out.close();    // close を忘れない
  }
}
```

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、（Web アプリケーションルート）/WEB-INF/というフォルダに置かれる。counter.txt の中身は 1 行のみの数字だけのファイルである。

プログラム中の

```
getServletContext().getRealPath( ... )
```

という式は、WEB アプリケーションルートからのパスを受け取り、ファイルシステム中の絶対パスを返す。getServletContext は HttpServlet クラスのメソッドであり、getRealPath は SevletContext クラス（正確にはインタフェース）のメソッドである。これらのメソッドの詳細は Java の API 仕様のドキュメント（例えば HttpServlet クラスの場合は、http://java.sun.com/j2ee/1.4/docs/api/javax/servlet/http/HttpServlet.html ）を参照すること。

また、

```
File f = new File(path);
BufferedReader fin = new BufferedReader(new FileReader(f));
...
fin.close();
```

は、ファイルから入力するときの常套句である。... の部分では、fin 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。最後の close() を忘れるとファイルの内容が消えてしまったりするので、注意が必要である。また、Integer.parseInt は Java で文字列（String）を整数（int）に変換するためのクラスメソッドである。（C 言語の atoi 関数に相当する。）

同様に、

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

は、ファイルへ出力するときの常套句である。fout というオブジェクトに対して、System.out（標準出力）に対するのと同じメソッドである print や println 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。

JDK 5.0 以降では、PrintWriter クラスに C 言語の printf 関数と同じような使い方ができる printf メソッドが用意されている。%n はプラットフォーム固有の改行コード（つまり、Unix では ¥n、Windows では ¥r¥n）を挿入する。

2.2 Java の例外処理

Counter.java では、ファイルからの入出力処理の周りを try〜catch〜という形で囲っているが、これは Java の例外処理の構文である。try〜catch 文のもっとも基本的な使い方は次のような形である。

```
try {
  文の並び0
} catch (例外型1 変数1) {
  文の並び1
}
   ...
  catch (例外型n 変数n) {
  文の並びn
}
```

文の並び$_0$ の中で、例外が起こった時には try { 〜 }の間の残りの文は無視され、例外が型$_k$（ただし k=1$\cdots$n）にマッチするならば、文の並び$_k$ が実行される。文の並び$_0$ で例外が起こらなかった場合、および例外が起こってもマッチする例外がなかった場合には、文の並び$_k$（k=1$\cdots$n）は実行されない。このプログラム例では、counter.txt というファイルが見つからなかった（FileNotFoundException）場合に、カウンタの値を 0 に設定している。

問 2.2.1 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。（割算の余りを求める演算子は、 *C* 言語と同様 %である。）

問 2.2.2 アプリケーションルートの *images* ディレクトリに、*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='../images/1.png'>, <img src='../images/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ができる。

数字を画像として表示するアクセスカウンタを作成せよ。

参考: 数字画像データ

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

問 2.2.3 数字を画像で表示する時刻表示プログラムを作成せよ。

2.3 （参考）ファイルを利用しない簡易アクセスカウンタ

Servlet のインスタンスは、ページのアクセス毎に生成されるのではなく、いったん生成されると、Tomcat の中で保持され 2 回目以降のアクセスでは、以前に生成された Servlet のインスタンスが再利用される。このため、インスタンス変数にデータを保持しておけば、ファイルを使用しなくても次のようなプログラムで簡易アクセスカウンタを実現できる。

ファイル Counter0.java

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

public class Counter0 extends HttpServlet {
  int i=0;          // インスタンス変数として宣言する

  @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>");
    out.printf("あなたは %d 番目の来訪者です。%n", i);
    out.println("</body></html>");
    out.close();    // close を忘れない
    i++;
  }
}
```

ただし、ファイルに書き込まないので、Tomcat を再起動すると、カウンタが 0 に戻ってしまう。完全なアクセスカウンタにするためには、Tomcat の終了時にカウンタの値をファイルに保存し、起動時にファイルからカウンタの値を読み込むように改造する必要がある。

問 2.3.1 *HttpServlet* クラスのメソッドを調べて、*Tomcat* 起動・終了時の操作を追加し、*Counter0* を完全なアクセスカウンタに改良せよ。

2.4 ディレクトリ操作

つぎの Servlet はあるディレクトリのインデックス（ファイルの一覧）を生成する。

ファイル DirIndex.java

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

public class DirIndex extends HttpServlet {
  @Override
  public void doGet(HttpServletRequest request, HttpServletResponse response)
              throws IOException {
    response.setContentType("text/html; charset=Windows-31J");
    PrintWriter out = response.getWriter();

    String path = getServletContext().getRealPath("/"); // 適切なパスに変えておく
    File dir = new File(path);
    String[] files = dir.list();  // dir にあるファイル名の配列を得る

    out.println("<html><head></head><body>");
    out.println("<pre>");

    int i;
    out.printf("%s のファイル一覧%n%n", path);
    for (i=0; i<files.length; i++) {
      out.println(files[i]);      // files の各要素を順に出力
    }

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

ディレクトリの中のファイル名の一覧は、File クラスの list メソッドで String の配列として得ることができる。また、配列の要素の数は length というメンバを調べることによってわかる。

問 2.4.1 *DirIndex.java* で、*3* 日前より変更された日付が新しいファイルには *"NEW!"* というマークをつけるようにせよ。例えば、ディレクトリに *old.txt* という *4* 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* 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>new.txt NEW!</li>
<li>old.txt</li>
</ul>
</body></html>
```

ヒント: *java.io.File* クラスの *lastModified* メソッドと *java.util.Date* クラスを用いる。

キーワード:

getServletContext メソッド, getRealPath メソッド, File クラス, FileReader クラス, BufferedReader クラス, FileWriter クラス, PrinterWriter クラス, 例外処理, try〜catch 文, list メソッド