第4章　セッション（session）の利用

4.1　セッションとは

Web サーバと Web ブラウザの間の通信 (HTTP) は本来一回ページを表示するごとに完結するものである。つまり、ユーザーが過去にどのようなページを見ていたか、などといった履歴には関係なく動作する。そこで、過去の履歴（例えば会員制ページのユーザのログイン情報や、ショッピングサイトの商品の選択の情報）に依存するような Web アプリケーションを実現するためには、なんらかの形で過去のユーザーのアクセスの情報を保持する必要がある。このようなデータは、Web サイトにアクセスするユーザごとに管理しなければならないので、単純にインスタンス変数やファイルなどに保存することはできない。 このようなユーザー毎の履歴データの保持には、クッキー（cookie）という技術を使う方法、フォームの隠し要素（hidden）を使う方法, URL 中の Query String に履歴データを埋め込む方法などがある。

問 4.1.1 クッキー（*cookie*）とは、どういう仕組みか、調べよ。

Java Servlet では**セッション（session）**[1]という仕組みを提供している。セッションは裏側ではクッキーなどの仕組みを使っているが、複雑な部分をプログラマが見なくても良いようにして、Servlet のプログラマが簡単にユーザーの履歴データを利用できるインタフェースを提供している。使い方は簡単で、`HttpServletRequest` クラスの `getSession` というメソッドでセッションオブジェクトを取り出し、そのオブジェクトに対して、データを関連づけたり、読み出したりするだけである。

4.2　Quiz サーブレット

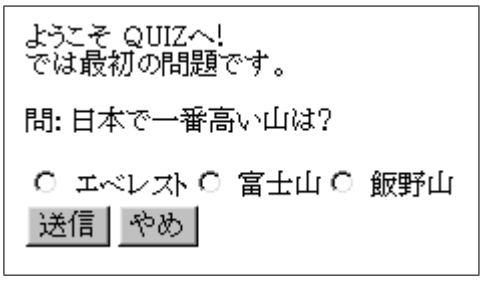
セッションを利用する簡単な Servlet として Quiz サーブレットを例にあげる。これは過去に正解した問題数などによって、表示を変更する Servlet である。

例題: QUIZ
この Servlet は次のようなテキストファイル
ファイル `quiz.txt`

```
日本で一番高い山は? エベレスト 富士山 飯野山 2
日本で一番広い湖は? 琵琶湖 府中湖 満濃池 1
トリノで日本が獲得した金メダルの個数は? 1個 2個 3個 1
工学部の所在地は? 幸町 花園町 林町 3
```

から次のような QUIZ を表示するページ

[1]もともとは会期などという意味



を作成する Servlet である。簡単のため問題は 3 択に固定している

この Servlet では “現在何番目の問題か？” (number) と “これまでの正解数”（score）というデータがセッションのなかに保持されている。

ファイル Quiz.java（その 1）

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

public class Quiz extends HttpServlet {
```

（その 1）の部分は特に変わったところはない。

ファイル Quiz.java（その 2）

```
  @Override
  public void doGet(HttpServletRequest request, HttpServletResponse response)
  throws IOException {
    doPost(request, response);
  }

  @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 number=0, score=0, answer=0;
    String message;
    File f = new File(getServletContext().getRealPath("/WEB-INF/quiz.txt"));
    BufferedReader in = new BufferedReader
    (new InputStreamReader(new FileInputStream(f), "Windows-31J"));
    String line="";
    int i;

    HttpSession session = request.getSession(true) ;
```

doGet メソッドは最初に CGI が実行される時に呼ばれる。このメソッドは単に doPost メソッドを呼び出すだけである。

doPost メソッドでは、いくつかの局所変数を宣言し、QUIZ のデータが入っているファイル（quiz.txt）を開く。そして、HttpServletRequest クラスの getSession メソッドで現在のセッションオブジェクトを得る。

ファイル Quiz.java（その 3）

```
    try { // 最初の問ではないとする
      answer = Integer.parseInt(request.getParameter("answer"));
      number = Integer.parseInt((String)session.getAttribute("number"));
      score  = Integer.parseInt((String)session.getAttribute("score"));
```

このページのアクセスが最初の問ではないとすると、セッションデータには現在の問題の番号（"number"）とこれまでの正解数（"score"）が記録されているはずなので、これを Session クラスの getAttribute メソッドでその値を取り出す。（最初の問の場合は、後述する。）getAttribute メソッドの引数は取り出したいデータの名前で、戻り値がセッションに保存されていたその名前のデータである。getAttribute メソッドの戻り値型は Object 型（つまりすべてのクラスのスーパークラス）なので、String 型に型変換（キャスト）する必要がある。

さらに、フォームのデータから、 前の問題で解答者が選んだ答の番号（answer）を得ている。

ファイル Quiz.java（その 4）

```
      for (i=0; i<number; i++) {
        line = in.readLine();  // number-1 行分、読み飛ばす
      }
      line = line.trim(); // trim() は前後の空白を除去する
      int a = Integer.parseInt(line.substring(line.length()-1));
      if (a==answer) { // a は最後の文字
        message = "正解です。<br>";
        score++;
      } else {
        message = "残念でした。<br>";
      }
    }
```

さらに quiz.txt を number-1 行分読み飛ばして、number 行目の最後の文字を解答と比べる。その結果によってメッセージと score 変数の値を変える。

一方、最初の問の場合には、セッションデータから getSession をしようとした段階で例外を発生するので、catch 節の中が実行される。

ファイル Quiz.java（その 5）

```
    catch (Exception ex) { // 最初の問だった
      message = "<p>ようこそ QUIZ へ!<br>では最初の問題です。</p>";
      number = 0; score = 0;
    }
```

ここではメッセージを最初の問にふさわしいものに設定している。

さらに、最初の問であろうとなかろうと、

ファイル Quiz.java（その 6）

```
    out.println(message);
    line = in.readLine();
    in.close();
    if (line==null || line.trim().equals("")) { // 終
      out.println("<br>これで QUIZ は終わりです。<br>");
      out.println("正解数は、"+score+"問でした。");
      session.invalidate();
    }
```

メッセージを出力したあとは、quiz.txt の次の行を読み込む。（line = in.readLine() ）line が

null か空文字列なら、そこで quiz.txt は終わりなのでクイズを終了する。invalidate メソッドはセッションオブジェクトの破棄を行う。

ファイル Quiz.java（その 7）

```
    else {
      StringTokenizer st1 = new StringTokenizer(line);
      out.println("次の問: "+st1.nextToken()+"<br>");
      out.println("<form method='post'>");
      for (i=0; i<3; i++) {
        out.print("<input type='radio' name='answer'");
        out.print(" value='"+(i+1)+"'>");
        out.print(" "+st1.nextToken());
      }
      out.println("<br>");
      out.println("<input type='submit' value='送信'>");
      out.println("<input type='reset' value='やめ'>");
      out.println("</form>");

      session.setAttribute("number", Integer.toString(number+1));
      session.setAttribute("score", Integer.toString(score));
    }
    out.println("</body></html>");
    out.close();
  }
}
```

終わりでなければ、line から次の問の情報を読み取って、フォームとして出力する。そして各選択肢のラジオボタンと送信ボタン、リセットボタンを出力する。

ここで、new StringTokenizer(line) は line という文字列の中の空白で区切られた単語を列挙したオブジェクト（StringTokenizer クラス）を返す。このクラスのオブジェクトに対しては、nextToken というメソッドで次の単語を取り出すことができる。また、ここでは使われていないが、hasMoreTokens というメソッドで単語がまだ残っているかどうかを調べることができる。

StringTokenizer クラスについては、(***(J2SEAPI)***/java/util/StringTokenizer.html) も参照すること。

最後に setAttribute メソッドを用いて、セッションオブジェクトにこれまでの問題数と正解数を記録している。setAttribute の引数は保存したいデータの名前（String 型）と保存したいデータ（Object 型）である。Integer.toString メソッドを使って、int 型から String 型に変換してから、setAttribute を呼び出して、セッションオブジェクトに書き込んでいる。（このデータは次の問題を表示するときに利用される。）

問 4.2.1 *Quiz.java* を *3* 択だけではなく、各問ごとに選択肢の数を変えられるように拡張せよ。

問 4.2.2 （選択肢の記録）
Quiz.java を、 正解数だけではなくて、どの問に対してどの選択肢を選んだかまでわかるように記録し、その結果を「これで *QUIZ* は終わりです。」のメッセージのあとに表示するようにせよ。

キーワード:

クッキー, hidden（隠し要素）, セッション, HttpSession クラス, getSession メソッド, getAttribute メソッド, StringTokenizer クラス, setAttribute メソッド, Integer.toString メソッド